2026 ASEE ANNUAL CONFERENCE June 21–24, 2026 | Charlotte Convention Center, NC

Host Sponsor UNC Charlotte

Paper ID #52301

Enhanced Private Ethereum Blockchain Education Environment (PE-BEE) and Laboratory Modules for Undergraduate Education

Dr. Emil H Salib, James Madison University

Professor in the College of Integrated Science & Engineering (CISE) at James Madison University (JMU). Current Teaching - Introductory Telecom and Networking, Advanced Networking & Advanced Network Security Current Research - Block Chain/Ethereum, Open-Source Private Cloud Computing in enterprise environments

Enhanced Private Ethereum Blockchain Education Environment (PE-BEE) and Laboratory Modules for Undergraduate Education

Dr. Emil H. Salib, salibeh@jmu.edu,
College of Science and Engineering (CISE),
James Madison University (JMU), Harrisonburg, VA 22807

1. Introduction and Background

This practice full paper represents on the design and implementation of a reproducible instructional environment, with emphasis on system transparency rather than on the evaluation of learning outcomes. Teaching blockchain technology at the undergraduate level continues to be challenging because students often cannot observe how blockchain systems behave internally [1]. Most instructional approaches rely on public networks or high-level development tools such as Remix, Ganache, and web-based contract interfaces. These tools allow students to deploy contracts and issue transactions, but they hide internal mechanisms such as genesis configuration, difficulty settings, peer discovery, and synchronization behavior. As noted in several studies, this lack of transparency reduces opportunities for students to build accurate conceptual models of consensus and distributed agreement [2, 3].

Ethereum, a widely studied blockchain platform, combines an account-based state model, peer-to-peer discovery protocols, and consensus mechanisms that historically relied on Proof-of-Work (PoW) [4]. Although extensive documentation exists, including the Yellow Paper and devp2p protocol specifications [5, 6], these documents are often difficult for undergraduates to understand without hands-on opportunities to observe system behavior.

The enhanced PE-BEE platform, the focus of this paper, was designed to address these gaps. It introduced: (1) an editable genesis file to allow experimentation with chain identity, difficulty, and initial allocations, (2) explicit separation of full/miner, light/client/account, and boot nodes, each running on its own virtual machine, (3) multi-network segmentation to encourage authentic peer discovery behavior, and (4) controlled conditions for capturing and interpreting devp2p discovery messages.

The goal of this paper is to contribute a practical and transparent undergraduate blockchain instruction environment that aligns with established blockchain and distributed-systems pedagogy [1, 2, 7, 8] by exposing genesis configuration, peer discovery, role-specific node behavior, and Proof-of-Work dynamics through reproducible laboratory modules. Unlike prior private blockchain teaching environments, the enhanced PE-BEE platform combines editable genesis configuration, explicit node-role separation, and multi-network peer-discovery segmentation within a single, resettable undergraduate laboratory framework.

2. Related Work

Smart-contract and application-focused teaching. A significant portion of the literature focuses on smart contracts. Eskandari et al. report that students enjoy writing simple contracts but often encounter conceptual difficulties when trying to understand how these contracts interact with

blockchain state and consensus [9]. Alharby and van Moorsel provide a mapping study of smart-contract platforms and highlight that educational tools tend to emphasize programming convenience over system transparency [10]. These works demonstrate the value of application-driven learning but also suggest that deeper system-level insight is often missing.

Earlier PE-BEE environment. The earlier PE-BEE platform [11] introduced a private Ethereum-based environment that supported contract deployment and simple transactions which aligned with literature focus at the time. However, that version did not expose genesis-parameter tuning, multi-network discovery behavior, or clear node-role differentiation.

Enhanced Private blockchain environments for teaching. To address limitations of public networks, several authors propose private or permissioned blockchain environments. Stokkink and Pouwelse describe controlled distributed-ledger experiments designed for teaching reliability and reproducibility [12]. Zhang et al. introduce containerized blockchain networks that allow instructors to scale experiments without managing many physical nodes [13]. While these environments help students observe basic interactions, they often use single-network layouts and may not separate node roles in a way that highlights Ethereum-specific internal behavior.

Peer discovery, networking protocols, and traffic inspection. Ethereum's devp2p discovery layer and RLPx transport protocol are documented in specifications [5, 6, 14] but are not easy for undergraduates to observe directly on public networks. Some research studies analyze blockchain traffic capture, but these are usually oriented toward security and protocol evaluation rather than undergraduate teaching [15, 16]. Chai et al. propose high-level visualization tools for Ethereum behavior, but these tools do not expose discovery packet flows [17]. The enhanced PE-BEE environment contributes a middle ground by enabling controlled capture of discovery behavior on a small but realistic private network.

Hands-on networking education. Networking education frequently uses tools such as Wireshark to help students connect observed protocol messages with conceptual descriptions [18]. The enhanced PE-BEE platform is consistent with these pedagogical approaches: it gives students visibility into node roles, message exchange, configuration dependencies, and synchronization, all within a manageable environment.

3. Motivation

Constraints of public networks. Public Ethereum networks impose fixed genesis parameters, allow no modification of chain identity, and cannot be reset for repeated experimentation. They also generate high volumes of encrypted traffic, which makes it difficult to observe devp2p discovery messages or block-synchronization behavior in a controlled way [5, 14]. As a result, students often interact with a running blockchain but do not see how it came into existence or how it maintains agreement among nodes. Prior work notes that without opportunities to observe these internal processes, student mental models of consensus and distributed coordination remain incomplete [3].

Gaps identified in the initial PE-BEE platform. The earlier PE-BEE environment used a simple single-network layout, did not separate full/miner, light/client/account, and boot/peer discovery nodes, and did not expose the effects of changing genesis parameters. There was no opportunity to examine discovery traffic or to understand how nodes locate peers. Feedback from that deployment indicated that students wanted more visibility into the "startup" behavior of the network, the meaning of the genesis file, and the role of configuration in successful block creation and synchronization.

Educational motivation. The educational need motivating the enhanced PE-BEE environment is therefore straightforward: students require a blockchain platform where internal mechanisms are visible, modifiable, and tied directly to hands-on laboratory work. The subsequent sections present the design and implementation of an environment that allows students to observe how configuration choices influence behavior, how nodes find peers, how blocks propagate, and how errors can prevent synchronization. Section 6 provides concrete laboratory modules that were developed and validated using that enivornment.

This paper presents an instructor-designed private Ethereum platform and associated laboratory framework, developed and validated with student participation, to support hands-on undergraduate instruction rather than to report formal learning assessment outcomes.

4. System Design and Architecture

The following sections describe the design, implementation, and validation activities carried out from an instructor perspective, with students contributing primarily through testing and execution of the developed laboratory modules and through providing feedback.

The enhanced PE-BEE platform was designed to provide a transparent, reproducible, and structurally accurate representation of how Ethereum nodes behave in a small private network. The design was guided by the following concrete requirements: (1) making internal blockchain mechanisms visible to students, and (2) keeping the environment simple enough to deploy and reset during typical undergraduate lab sessions.

Overall design objectives. Four design objectives guided the enhanced PE-BEE platform. First, the environment needed to expose configuration details that are normally hidden in public networks, especially genesis parameters and node-launch settings. Second, node roles had to be separated so that students could observe how full/miner, light/client/account, and boot/peer discovery nodes behave differently in practice. Third, the network itself needed to promote realistic peer-discovery behavior without overwhelming students with unnecessary complexity. Fourth, a minimal Ethereum PoW instructional environment requires one discovery-only bootnode, at least two full-node miners to enable observable block competition, and one or more light-node clients A client node could manage multiple accounts that generate transactions.

Editable genesis configuration

A central design component was the use of an editable `genesis.json` file. This file allowed instructors and students to modify: (1) the chain identifier, (2) initial account allocations, (3) gas limits, and (4) the Proof-of-Work (PoW) difficulty. Ethereum's behavior at genesis is well documented in the Yellow Paper and related protocol descriptions [4]. However, public networks do not permit modification of these parameters. In the current project, editable genesis files allowed students to see how changes influence block creation rate, chain identity, and node compatibility. Several baseline genesis templates were validated by the instructor before being distributed to students.

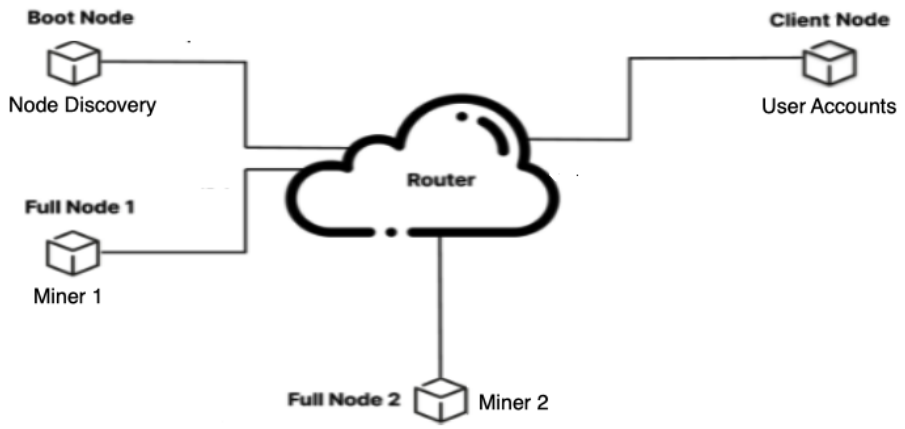

Figure 1: Enhanced PE-BEE Design Network Diagram

Ethereum blockchain system nodes

Ethereum distinguishes full/miner, light/client/accounts, and boot/peer discovery nodes based on capabilities, responsibilities, and synchronization behavior [4]. To highlight these distinctions, the enhanced PE-BEE Platform allocated each role to its own node.

- **Full** nodes maintained the full chain state and synchronized new blocks.
- **Full/Miner** nodes ran PoW mining and produced blocks at the configured difficulty.
- **Light (Client/Accounts)** nodes synchronized only headers and validated block summaries.
- **Boot (Peer Discovery) node** provided initial peer records using the devp2p discovery protocol.

This separation allowed students to observe role-specific logs, peer counts, and synchronization behaviors. It also aligned with descriptions in the Ethereum documentation and research studies on node diversity [14, 15].

At this point, it is important to note that authentication in the Ethereum blockchain system occurs at three distinct layers: account authentication via ECDSA transaction signatures, client authorization via local key custody and JSON-RPC submission, and peer authentication via RLPx session establishment.

Multi-network topology

As shown in Figure 1, the enhanced PE-BEE platform was designed to use several separate LAN networks to encourage authentic Ethereum peer-discovery behavior. Nodes could not rely on broadcast discovery within a single LAN; instead, they used Ethereum's devp2p discovery protocol to find peers across network boundaries. This design reflected the documented behavior of Ethereum nodes in real deployments [5, 6].

The multi-network topology was intentionally minimal, where each LAN serves one or more of the following nodes: boot/peer discovery, light/client/account and full/miner. This structure not only supported discovery-based connection formation but also helped students reason about which messages crossed which boundaries.

Alignment with Ethereum specifications

The enhanced PE-BEE platform was designed to remain pedagogically simple while still maintaining alignment with documented Ethereum behavior. Several design decisions reflected this goal:

- Genesis configuration conformed to the field definitions and state-transition semantics specified in the Ethereum Yellow Paper [4].
- Node startup and runtime parameters followed the official go-ethereum (geth) documentation to ensure canonical client behavior [19].
- Peer discovery used the devp2p discovery protocol (PING, PONG, FINDNODE, NEIGHBORS) as specified in the protocol definitions [5, 6].
- Transport-layer communication employed RLPx with encrypted session establishment and message framing consistent with published specifications [14].
- Transaction submission was performed by a wallet client (MetaMask), operating as a non-peer client node via JSON-RPC, with local signing and submission through standard methods (e.g., eth_sendRawTransaction) and no participation in devp2p discovery or block gossip [20, 21].

Although the enhanced PE-BEE environment was small in scale, it reproduced core Ethereum mechanisms accurately enough for students to make meaningful observations about consensus, peer formation, and synchronization.

Reproducibility and reset workflows

Undergraduate labs require platforms that can be reset easily. The enhanced PE-BEE design incorporated: version-controlled genesis files, startup scripts for each node role, simple commands for clearing chain data, and validation steps for checking node compatibility. These workflows, ensured that an instructor could rebuild the environment quickly if errors occurred or if multiple student groups needed to start from a clean state.

5. Platform Implementation

The implementation of the enhanced PE-BEE platform focused on constructing a concrete, reproducible private Ethereum environment aligned with the design goals presented in Section 4. Implementation tasks included provisioning virtual machines (VMs), configuring networks, preparing genesis files, validating node behavior, and developing supporting scripts.

Virtual machine provisioning

The instructor provisioned a set of Linux VMs, each assigned a specific role: full/miner, light/client/account, or bootnode. Each VM included: a consistent Linux distribution, an installed and validated version of the go-ethereum (`geth`) client [19], and essential administrative tools such as log viewers and network utilities. VM images were stored so that the environment could be reproduced easily for additional student groups or future semesters.

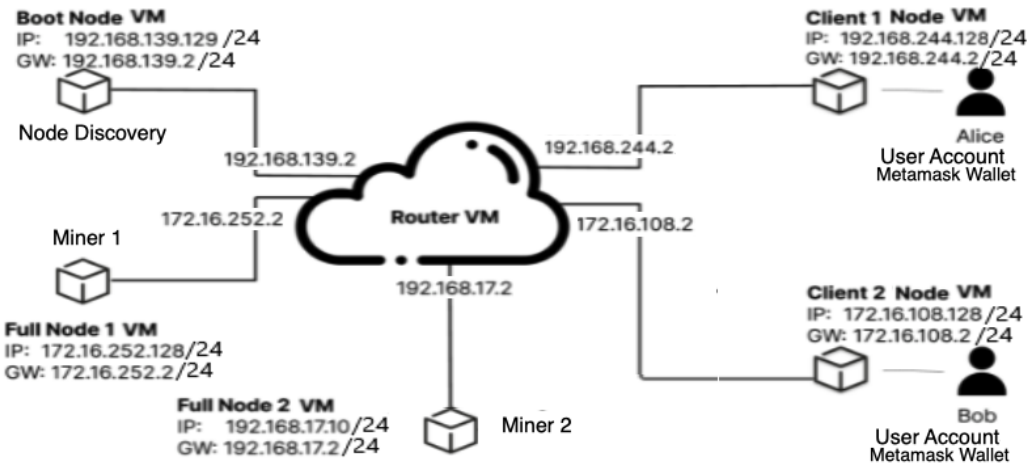

Figure 2: Enhanced PE-BEE Implementation Network Diagram

Network configuration

Each virtual machine (VM) was connected to one virtual network except the router VM as shown in Figure 2. The addressing scheme was intentionally simple so that students could track communication among nodes. This configuration supported authentic peer-discovery behavior because nodes could not rely on broadcast discovery within a single LAN.

Genesis file preparation

The instructor prepared a baseline `genesis.json` file containing a private chain identifier, prefunded accounts, a suitable gas limit for laboratory transactions, and a PoW difficulty low enough to allow observable block creation on lab hardware. allow observable block creation on lab hardware.

Node initialization

Before any laboratory activity, each node was initialized using: `geth init genesis.json`. This command created the initial database for each node, ensuring that all nodes shared the same genesis state. The instructor validated that the generated genesis hash was the same on all nodes, no stale chain data remained, and each node was ready to join a fresh network. These validation steps were crucial, as mismatched genesis files prevent nodes from forming a common chain—a concept echoed in the blockchain literature [15, 16].

Role-specific startup scripts and validation

Each node role required a distinct set of startup parameters. For example: full nodes synchronized block and state data, full/miner nodes used flags enabling PoW mining and specifying the mining account, light/client/account nodes synchronized headers only, and the bootnode used the devp2p discovery service to advertise peer information [5, 6]. To reduce student errors and simplify resets, the instructor created a set of role-specific scripts. Before releasing the laboratory modules to students, the instructor performed detailed trial runs confirming the initial state and operational integrity of the platform.

Overall, the implementation phase ensured that the enhanced PE-BEE environment was stable, predictable, and sufficiently transparent to support the laboratory modules described in Section 6.

6. Laboratory Modules

The enhanced PE-BEE environment was used to develop a sequence of hands-on laboratory exercises. This effort resulted in the design, development, testing, and delivery of five laboratory modules. Each module was designed to help students observe internal blockchain behavior that is usually hidden in public networks. Across the modules, students performed configuration tasks, launched nodes, inspected logs, and diagnosed mis-configurations. The sequence reflected recommendations in distributed-systems such as blockchain systems education for giving students clear opportunities to observe both correct and incorrect system behavior [7, 8].

Lab 1: Genesis exploration and chain initialization

The first lab introduced students to the genesis configuration. Students examined the instructor-provided `genesis.json` file and identified key fields such as the chain identifier, gas limit, allocations, and PoW difficulty. Students made minor controlled changes—such as adjusting the difficulty or modifying an allocation—to observe how these changes influenced subsequent block creation and compatibility among nodes.

After editing the genesis file, students initialized their nodes. They confirmed that nodes sharing the same genesis file could join the same chain, while nodes with mismatched genesis files failed to connect. This activity demonstrated the importance of chain identity, as discussed in Ethereum documentation and prior literature on consensus and blockchain state [4, 15].

Lab 2: Node roles and network formation

The second lab focused on node-role behavior and peer formation. Students launched full/miner, light/client/account, and boot/peer discovery nodes using instructor-prepared startup scripts. They used console logs to observe peer attempts, discovery messages, peer-table updates, and synchronization progress. The role separation allowed students to connect concrete node behaviors to descriptions from Ethereum documentation [4] and discovery-protocol specifications [5, 6].

Students also monitored metrics such as peer count and synchronization status. Observing nodes discovering peers across multiple network segments helped students understand why Ethereum uses the devp2p protocol suite and how discovery differs from simple LAN broadcast techniques.

Lab 3: Transactions, mining, and block validation

The third lab explored authentication (account, client and peer), transaction flow, mining, validation and PoW consensus algorithm (see Figure 3). Students submitted transactions between accounts (established on different client nodes as shown in Figure 2) and observed how these arrived in node mempools before being included in blocks. They observed how mining rate varied with the configured difficulty and how block intervals changed when difficulty was adjusted within a safe range.

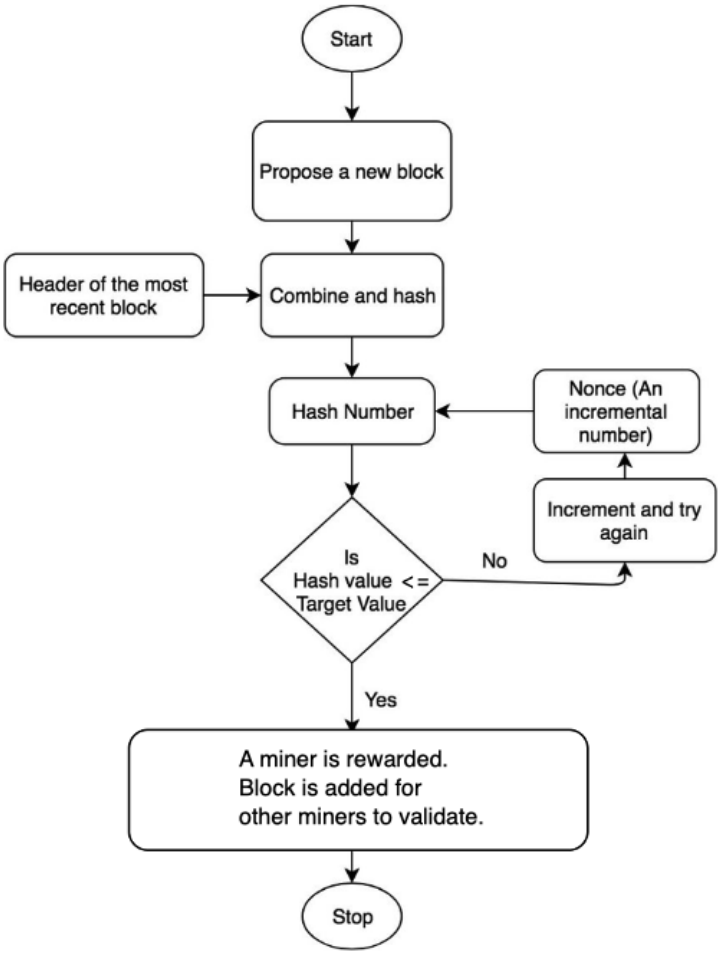

Figure 3: PoW Consensus Algorithm Flow

These experimentations helped students connect abstract explanations of PoW [22, 23] with the concrete behavior of transaction processed in a running Ethereum environment. Observing the relationships among mining rate, transaction confirmation time, and block validity reinforced concepts discussed in prior analyses of blockchain consensus [16].

Lab 4: Peer discovery and protocol behavior

The fourth lab focused on peer-discovery behavior. Although the PE-BEE environment used only a basic packet exchange traffic dissector, students were still able to observe discovery-related activity through console logs and non-encrypted packet captures. They examined patterns that corresponded to PING, PONG, FINDNODE, and NEIGHBORS messages [5, 6]. Students compared these observations with documentation of Ethereum's discovery protocol and explanations of the RLPx transport layer [14].

This lab helped students relate low-level discovery activity to the process of forming the peer table and establishing encrypted channels—topics that are usually inaccessible when working with public Ethereum networks.

Lab 5: Debugging and recovery

The final lab emphasized debugging and recovery. Students were provided with intentionally misconfigured scenarios, such as: mismatched genesis files, incorrect bootnode addresses, stale chain databases, or nodes launched with incorrect role-specific parameters.

Students analyzed logs, packet captures (if available), identified configuration errors, and applied corrective actions. This activity highlighted the importance of observing system state carefully, understanding dependencies among configuration files, and reasoning about how node-startup parameters influence behavior. The approach aligns well with distributed-systems such as blockchain system education practices that emphasize debugging as a critical learning objective [7, 8].

7. Discussion, Conclusion and Future Work

Discussion

Although the enhanced PE-BEE platform is small in scale and limited to a controlled laboratory environment, its design demonstrates how internal blockchain mechanisms can be made observable and pedagogically meaningful for undergraduate learners.

Position within blockchain education. Much of the existing educational literature focuses on teaching smart contracts and high-level application topics [9,10,17]. These resources are important, but they often leave internal mechanisms—such as genesis initialization, peer discovery, mining behavior, and block validation—hidden from students.

The ability provided by the enhanced PE-BEE to experiment with genesis parameters, difficulty values, and role-specific node behavior helps address several educational challenges noted in prior work [1–3]. Students can explore not only what blockchains do, but why they behave as they do in response to specific design and configuration decisions.

Pedagogical impact of internal visibility. For example: mismatched genesis files cause nodes to reject each other, discovery messages populate the peer table gradually, difficulty settings influence block intervals noticeably on small hardware, and light nodes behave differently from full nodes during synchronization.

Informal indicators of student learning. Students initially treated Ethereum as an opaque system, viewing mining and block formation as background abstractions. Through direct interaction with genesis configuration, block structure, node roles, and synchronization workflows, they developed clearer understanding of how transactions are assembled, validated, and committed to the chain. In the Proof-of-Work module, examining the miner, worker, agent, and PoW engine interactions—along with tracing and simulating portions of the Ethash algorithm—helped students recognize mining as a coordinated algorithmic process rather than a single function. Informal reflections frequently noted improved clarity regarding nonce search, difficulty effects, workflow dependencies, and diagnosis of synchronization or mining misconfigurations.

Conclusion

By exposing editable genesis configuration, separating node roles, employing a multi-network topology, and enabling controlled observation of peer discovery and synchronization behavior, the enhanced PE-BEE environment addresses instructional gaps identified in earlier deployments and in the broader blockchain education literature [1–3]. The platform remains aligned with documented

Ethereum mechanisms, including genesis initialization, Proof-of-Work block production, devp2p-based peer discovery, and role-specific synchronization behavior [4–6, 14].

Although intentionally limited in scale, the environment reproduces core Ethereum behaviors with sufficient fidelity to support meaningful undergraduate engagement with internal protocol dynamics. Instructor validation and student execution confirmed consistent behavior across laboratory deployments, with observations suggesting improved clarity regarding configuration dependencies, discovery processes, block propagation, and debugging workflows.

The enhanced PE-BEE platform strikes a deliberate balance between technical fidelity and classroom manageability. It offers educators a reproducible framework for introducing blockchain not only as an application platform, but as a distributed system whose internal mechanisms can be observed, modified, and reasoned about directly. This work contributes a concrete, technically grounded instructional model that can be extended to support additional protocol visibility, alternative consensus mechanisms, and future blockchain teaching innovations.

Future Work

Future development will focus on deepening protocol visibility, increasing flexibility, and expanding the platform's relevance to contemporary blockchain systems.

- Integration of the advanced Wireshark dissector
- Scaling and containerization of the PE-BEE environment
- Support for additional consensus mechanisms
- Security-focused laboratory modules and extensions

Acknowledgment

The author would like to thank the College of Science and Engineering (CISE) and the Computer Science (CS) Department at James Madison University (JMU) for their support throughout the development and creation of this paper. Special thanks to the Spring 2022 project team members (students), Chenguang Cao, Boyuan Gou and Matthew McCreary who contributed to the design, configuration, validation, testing and documentation of the enhanced PE-BEE environment.

References

[1] Z. Chen, Y. Xu, and S. Yang, "Teaching blockchain systems with hands-on experimentation: Lessons from undergraduate courses," *IEEE Access*, 2024.

[2] M. Natarajan and S. Dharmaraj, "Blockchain education in computing programs: A systematic review," *Education and Information Technologies*, 2023.

[3] H. Wang, A. Cepeda, and T. Quek, "Experiences with teaching blockchain fundamentals using practical exercises," *Journal of Computing Sciences in Colleges*, 2021.

[4] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," https://ethereum.github.io/yellowpaper/, 2022.

[5] Ethereum Foundation, "Devp2p wire protocol specification," https://github.com/ethereum/devp2p, 2019.

[6] V. Buterin *et al.*, "Ethereum node records (enr), EIP-778," https://eips.ethereum.org/EIPS/eip-778, 2019.

[7] Y. Brun, F. E. Bustamante, and M. D. Ernst, "Debugging distributed systems: Challenges and opportunities for education," in *Proceedings of the ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE)*, 2021.

[8] P. Potocnik, B. Novak, and T. Kosar, "Challenges of teaching distributed systems: A survey," *IEEE Transactions on Education*, 2020.

[9] S. Eskandari, D. Barrera, E. Stobert, and J. Clark, "A first look at teaching smart contracts: Experiences and challenges," in *Proceedings of the IEEE Frontiers in Education Conference*, 2019.

[10] M. Alharby and A. van Moorsel, "Blockchain-based smart contracts: A systematic mapping study," *Proceedings of the IEEE International Conference on Cloud Computing Technology and Science*, 2017.

[11] Anonymous, "A private platform for teaching blockchain at the undergraduate level," in *Proceedings of the ASEE Annual Conference*, 2021.

[12] Q. Stokkink and J. Pouwelse, "Teaching distributed ledger technology with controlled experiments," in *Proceedings of the EduPar Workshop*, 2020.

[13] Y. Zhang, Z. Li, and H. Chen, "Hands-on blockchain labs with containerized networks," in *Proceedings of the ACM Special Interest Group for Information Technology Education (SIGITE)*, 2022.

[14] P. Szilágyi, "Rlpx: Cryptographic transport protocol," https://github.com/ethereum/devp2p/blob/master/rlpx.md, 2020.

[15] J. Bonneau *et al.*, "Sok: Research perspectives and challenges for bitcoin and cryptocurrencies," *Proceedings of the IEEE Symposium on Security and Privacy*, 2015.

[16] C. Cachin and M. Vukolić, "Blockchain consensus protocols in the wild," *arXiv preprint arXiv:1707.01873*, 2017.

[17] M. Chai, K. Lam, and J. Li, "Understanding ethereum blockchain through simulation and visualization tools," *International Conference on Educational Technology (iCET)*, 2020.

[18] Y. A. Lee and D. Klappholz, "Teaching protocol analysis with wireshark in computer networking courses," in *Proceedings of the ASEE Annual Conference*, 2018.

[19] Ethereum Foundation, "go-ethereum (geth) documentation," https://geth.ethereum.org/docs/, 2021.

[20] MetaMask, "MetaMask Documentation: Json-rpc usage and local transaction signing," https://docs.metamask.io/, 2026, accessed: 2026-01-15.

[21] Ethereum Foundation, "Ethereum Execution APIs: Json-rpc specification (`eth_sendRawTransaction`)," https://ethereum.org/en/developers/docs/apis/json-rpc/, 2026, accessed: 2026-01-15.

[22] Y. Sompolinsky and A. Zohar, "Secure high-rate transaction processing in bitcoin," in *Financial Cryptography and Data Security*, 2015.

[23] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," https://bitcoin.org/bitcoin.pdf, 2008.