UWL REPOSITORY

repository.uwl.ac.uk

Blockchain security using confidentiality, integrity, and availability for secure communication

Francis Ikenga-Metuh, Chukwuebuka and Yeboah-Ofori, Abel ORCID logoORCID: https://orcid.org/0000-0001-8055-9274 (2026) Blockchain security using confidentiality, integrity, and availability for secure communication. Blockchains, 4 (1).

https://doi.org/10.3390/blockchains4010003

This is the Published Version of the final output.

UWL repository link: https://repository.uwl.ac.uk/id/eprint/14685/

Alternative formats: If you require this document in an alternative format, please contact: open.research@uwl.ac.uk

Copyright: Creative Commons: Attribution 4.0

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy: If you believe that this document breaches copyright, please contact us at open.research@uwl.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

Rights Retention Statement:

blockchains

Article

Blockchain Security Using Confidentiality, Integrity, and Availability for Secure Communication

Chukwuebuka Francis Ikenga-Metuh and Abel Yeboah-Ofori *

School of Computing and Engineering, University of West London, St. Mary's Road, Ealing, London W5 5RF, UK; ck@iksectors.co.uk or 32110022@student.uwl.ac.uk

* Correspondence: abel.yeboah-ofori@uwl.ac.uk

Abstract

Background: Blockchain technology has emerged as a transformative communication solution for securing distributed systems. However, several vulnerabilities exist during transactions, including latency and network congestion issues during mempool processing, topology weaknesses, cross-chain bridge exploits, and cryptographic weaknesses. These vulnerabilities have led to attacks that have threatened system integrity, including Block Extractable Value (BEV) attacks, Maximal Extractable Value (MEV) attacks, sandwich attacks, liquidation, and Decentralized Finance (DeFi) reordering attacks, among others. Thus, implementing a robust security framework based on the Confidentiality, Integrity, and Availability (CIA) triad remains critical for addressing modern blockchain technology threats. **Objective**: This paper examines blockchain technology, its various vulnerabilities, and attacks to determine how criminals exploit the system during transactions. Further, it evaluates its impact on users. Then, implement a blockchain attack in a "MasterChain" virtual environment to demonstrate how vulnerable spots can be practically exploited and discuss the application of the CIA security triad through modern cryptographic primitives. **Methods**: The approach considers Hevner's design science framework, which emphasizes creating innovative artifacts that address identified problems while contributing to the knowledge base through rigorous evaluation. Furthermore, we developed a MasterChain tool using Python with Flask for distributed node communication, utilizing the Elliptic Curve Digital Signature Algorithm (ECDSA) with the Standards for Efficient Cryptography Prime 256-bit Koblitz curve 1 (secp256k1) for digital signatures and Secure Hash Algorithm 3 (SHA-3) (Keccak-256) hashing for block integrity. **Results**: show how the CIA has been implemented to provide secure communication through ECDSA-based transactions, SHA-3 chain integrity verification, and a multi-node distributed architecture, respectively. The performance analysis shows that ECDSA provides 256-bit security with 64-byte signatures compared to 2048-bit Rivest–Shamir–Adleman (RSA)'s 256-byte signatures, achieving a 75% reduction in bandwidth overhead. SHA-3 provides immunity to length extension attacks while maintaining equivalent collision resistance to SHA-256. **Conclusions**: The MasterChain framework provides a practical foundation for implementing blockchain security that addresses both classical and emerging vulnerabilities. The adoption of ECDSA and SHA-3 (Keccak-256) positions the system favourably for modern blockchain applications, while providing insights into the cryptographic trade-offs between performance, security, and compatibility.

Academic Editor: Hamed Taherdoost

Received: 15 August 2025
Revised: 25 February 2026
Accepted: 26 February 2026
Published: 28 February 2026

Copyright: © 2026 by the authors. Submitted for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license.

Keywords: blockchain security; CIA triad; ECDSA; SHA-3; Keccak-256; MEV attacks; distributed ledger technology; cryptographic hash functions; smart contracts; Cybersecurity

1. Introduction

1.1. Background and Context

Blockchain technology and security continue to revolutionize the way data is handled during network communication, from enhancing transparency to improving security. [1] and has evolved through a distributed computing paradigm with applications across cloud, finance, healthcare, supply chain management, and governance. The fundamental promise of blockchain, decentralized trust through cryptographic verification, has made it attractive for scenarios requiring tamper-evident record-keeping without centralized authorities [2]. However, as blockchain adoption accelerates, so does the sophistication of attacks against these systems. Recent analyses reveal that blockchain platforms have been compromised for over USD 2.8 billion through cross-chain bridge exploits alone, representing approximately 40% of all Web3 security incidents [3]. Additionally, MEV attacks have extracted over USD 5 billion from Ethereum and related networks through sophisticated transaction manipulation schemes [4]. These vulnerabilities underscore the critical need for robust security frameworks grounded in established information security principles that ensure CIA. Several vulnerabilities exist during transactions, including latency and network congestion issues during mempool processing, topology weaknesses, cross-chain bridge exploits, and cryptographic weaknesses. These vulnerabilities have led to attacks that have threatened system integrity, including BEV attacks, MEV attacks, sandwich attacks, liquidation and DeFi reordering attacks, among others.

Prior studies have applied the CIA triad to the blockchain security conceptually [5], and architecturally in Cyber–Physical Systems (CPS) and Internet of Things (IoT) environments [6]. Recent blockchain research has increasingly emphasized confidentiality and trust mechanisms [7]. However, the practical enforcement of CIA principles within blockchain communication remains insufficiently explored.

1.2. The CIA Triad in Blockchain Context

The CIA triad represents the cornerstone of information security practice [8]. In blockchain technology, these principles manifest uniquely when not secure, allowing threat actors to gain unauthorized access to blockchain systems, manipulate transactions, and alter data, leading to information disclosure and data distortion. Thus, making information untrustworthy, unavailable, and unsafe to use during blockchain transactions. Confidentiality ensures that transaction data and user identities remain protected from unauthorized disclosure throughout the authentication, authorization, and transaction validation processes. While blockchain's transparency is a feature, selective confidentiality through cryptographic techniques enables privacy-preserving transactions without sacrificing verifiability [9]. Integrity guarantees that once data is recorded in the blockchain, it cannot be altered retroactively without detection, ensuring non-repudiation and accountability through proper audit trails. One of the key integrity mechanisms is the use of cryptographic hashing functions to create immutable chains, making any tampering immediately evident to all network participants [10]. Availability ensures that the blockchain network remains accessible and operational during distributed operations and consensus processing despite node failures, network partitions, or malicious attacks. Availability ensures that distributed replication across multiple nodes provides redundancy and fault tolerance during blockchain node communications [11].

1.3. Blockchain Challenges

Current blockchain implementations face several critical challenges that compromise their performance and security standards when analyzed for efficiency and safety during transaction executions. The paper focuses on the more prominent challenges that affect blockchain systems, including cryptographic efficiency, hash function vulnerabilities, and emerging threats.

Cryptographic Efficiency: Traditional cryptographic systems often employ RSA-based digital signatures, which require large keys (2048–4096 bits) to achieve adequate security. However, modern blockchain systems have adopted more efficient alternatives. Both Bitcoin [1] and Ethereum [12] employ ECDSA with 256-bit keys, achieving security levels equivalent to 3072-bit RSA [13] while dramatically reducing bandwidth overhead and processing delays in high-transaction environments [12,14].

Hash Function Vulnerabilities: Traditional implementations using SHA-256 inherit the Merkle–Damgård construction, which is susceptible to length extension attacks unless additional protective measures, such as Hash-based Message Authentication Code (HMAC), are employed. This vulnerability requires careful implementation to maintain integrity guarantees [15].

Emerging Threat Landscape: Modern blockchain systems face sophisticated attacks, including MEV extraction through transaction ordering manipulation [4]. A recent study quantifying BEV, a broader term encompassing all forms of extractable value, reveals the magnitude of this threat: over USD 540.54 million was extracted across 32 months, with sandwich attacks alone constituting 51.56% (USD 289.76 million) of the total extraction [16]. These attacks not only generate financial losses but also fundamentally threaten consensus security, with research demonstrating that rational miners will fork the blockchain when BEV opportunities exceed four times the block reward [16]. Additionally, cross-chain bridge exploits targeting protocol weaknesses have resulted in USD 2.8 billion in losses [3], representing approximately 40% of all Web3 security incidents. The convergence of these threats, BEV exploitation, bridge vulnerabilities, and consensus-layer weaknesses, demonstrates that existing frameworks inadequately address the multi-dimensional security challenges facing contemporary blockchain systems [16,17].

Quantum Cryptographic Threats: The quantum computing era poses existential threats to current asymmetric encryption attacks and hash-based symmetric encryption attacks. Systems employing RSA and traditional discrete logarithm-based schemes face vulnerabilities to Shor's algorithm, necessitating proactive migration strategies [16]. This Shor's algorithm challenges have raised several concerns relating to quantum-resistance threats and quantum-save issues on organizational systems, such as exploits on ECDSA signatures and TLS certificates during fork transactions, where the attacker and the legitimate user find blocks at almost the same time, as the attacker leading to the attacker deploying forged digital signatures.

2. State of the Art

This section discusses the state-of-the-art and literature review of blockchain security, technology, its architecture, components, and various security systems. Blockchain systems comprise several essential elements. The distributed Ledger provides a replicated database maintained across multiple nodes, which provides redundancy and eliminates single points of failure [2]. Each node maintains a complete copy of transaction history, enabling independent verification. The Cryptographic Hash Functions: One-way functions that map arbitrary-length input to fixed-length output, creating tamper-evident fingerprints of data. Bitcoin and Ethereum employ SHA-256 and Keccak-256, respectively, though the choice of hash function significantly impacts system properties [18].

Digital Signatures: Asymmetric cryptography enables transaction authorization without revealing private keys. Bitcoin utilizes ECDSA with the secp256k1 curve, while newer blockchains explore alternatives, including Edwards-curve Digital Signature Algorithm (EdDSA) and Schnorr signatures [19].

Consensus Mechanisms: Protocols enabling distributed nodes to agree on blockchain state despite network asynchrony and potential malicious actors. Proof-of-Work (PoW), Proof-of-Stake (PoS), and Byzantine Fault Tolerance (BFT) variants each present distinct security and performance trade-offs [20].

2.1. Application to Blockchain Security

Regarding the application of blockchain security, ref. [10] analyzes the CIA triad in relation to blockchain technology in public sector contexts. The framework demonstrated how distributed ledgers address each of the CIA components while introducing unique challenges. Ref. [11] presented a blockchain-based authentication system implementing CIA principles for e-health systems, demonstrating lightweight cryptographic techniques with 0.059 millisecond authentication delays, a 4000-fold improvement over contemporary approaches, while preserving CIA properties through Proof of Authority consensus. The authors integrated Ethereum blockchain, smart contracts, blind signatures, and one-way hash functions to enhance data integrity, security, and privacy in decentralized healthcare frameworks. Ref. [21] explored cryptography algorithms in a network environment by applying secret keys, public keys, and hash functions to ensure data confidentiality, authentication, and non-repudiation. Ref. [22] carried out a systematic review that examined blockchain vulnerabilities across network-level attacks, consensus-based exploits on smart contract vulnerabilities, and user-centric risks, including DeFi exploits, flash loan attacks, and how bridge vulnerabilities challenge traditional CIA assumptions.

2.2. Cryptographic Primitives in Blockchain

Cryptographic Primitives in blockchain provide the low-level mathematical algorithms used to provide secure CIA mechanisms in blockchain systems, including Digital Signature Algorithms, RSA vs. ECDSA. Furthermore, ref. [20] examined how digital signatures facilitate authentication and non-repudiation in blockchain systems, thereby enabling transaction authorization without the need for centralized authorities. Two dominant approaches, RSA and ECDSA, present distinct trade-offs. RSA relies on the computational difficulty of factoring large composite numbers. Ref. [23]. RSA's security derives from the RSA problem: given n = pq (the product of two large primes, p and q), and e (the public exponent), computing d (the private exponent) without knowing p and q is computationally infeasible. However, ECDSA bases security on the discrete logarithm problem in elliptic curve groups [24]. RSA signatures require key sizes of 2048–4096 bits for adequate security, resulting in 256–512-byte signatures. Given point P = kG on an elliptic curve (where G is a generator point and k is a scalar), determining k is computationally demanding. ECDSA achieves equivalent security to RSA with dramatically fewer keys: 256-bit ECDSA provides security comparable to 3072-bit RSA [13]. Comparative Analysis provides recent literature that demonstrates ECDSA's superiority for blockchain applications across multiple dimensions:

Key and Signature Size: ECDSA generates 64-byte signatures (for secp256k1) versus 256-byte RSA-2048 signatures, reducing blockchain storage and bandwidth requirements by 75% [14,25]. This difference compounds significantly in high-transaction environments. Ethereum processes approximately 1.2 million transactions daily, where ECDSA's compact signatures save around 230 MB daily compared to an equivalent RSA implementation.

Performance Characteristics: ECDSA signature generation is substantially faster than RSA signing. Benchmarks on contemporary hardware show ECDSA signing at ~0.2 milliseconds, compared to RSA-2048 at ~1.8 milliseconds [26]. However, RSA verification (0.16 ms) outperforms ECDSA verification (8.53 ms) due to the use of small public exponents (typically e = 65,537), which enables efficient modular exponentiation. [27].

Blockchain Adoption: Bitcoin, Ethereum, and the majority of modern blockchains employ ECDSA rather than RSA [12,14,26]. This industry consensus reflects ECDSA's practical advantages: reduced transaction size, enabling more transactions per block; lower bandwidth consumption for network propagation; and better alignment with the blockchain's peer-to-peer architecture, which requires frequent signature operations.

Security Considerations: Both RSA and ECDSA algorithms are vulnerable to quantum-resistant threats such as side channel attacks, integer factorization attacks, if the attacker could find the modulus *N* used to calculate the private keys in the RSA algorithm, as well as Denial of Service attacks, among others. Further, concerns are raised regarding quantum-resistance challenges, as it is able to resist threats but will not be able to prevent quantum attacks completely. Additionally, these algorithms may not be able to resist sufficiently powerful quantum computers and could allow threat actors to duplicate signatures, gain access, and modify content during blockchain transactions, leading to quantum-safe issues. For instance, Shor's algorithm can break RSA and ECDSA with sufficiently powerful quantum computers [16]. However, ECDSA's current classical security advantage and smaller footprint position it favourably for hybrid post-quantum transition strategies [28].

2.3. Hash Functions: SHA-256 vs. SHA-3 (Keccak)

Cryptographic hash functions form the backbone of blockchain integrity, creating collision-resistant fingerprints that link blocks into immutable chains. The choice between SHA-256 (SHA-2 family) and SHA-3 (Keccak) has a significant impact on system security and performance. SHA-256 (SHA-2 Family) was standardized by the National Institute of Standards and Technology (NIST) in 2002 as part of the Secure Hash Standard [29]. It employs the Merkle–Damgård construction with Davies–Meyer compression function, processing 512-bit blocks through 64 rounds of ARX (Add-Rotate-XOR) operations to produce 256-bit digests. SHA-256 demonstrates strong security properties with no practical collision attacks discovered to date [18]. SHA-3 (Keccak-256) represents a significant departure from the design philosophy of SHA-2. Selected as NIST's SHA-3 standard in 2013 after a five-year competition, Keccak employs the sponge construction, a permutation-based approach fundamentally different from Merkle–Damgård [30]. The sponge function absorbs input bits into a state through Exclusive OR (XOR) operations, applies the Keccak-f permutation, and squeezes output bits, providing flexible output lengths and inherent resistance to length extension attacks.

Cryptographic Diversity: SHA-3's entirely different design from SHA-2 provides crucial cryptographic diversity. If cryptanalytic breakthroughs compromise the Merkle–Damgård construction (as occurred with Message Digest Algorithm (MD5) and SHA-1), SHA-3 remains unaffected due to its fundamentally distinct sponge architecture [30,31]. This "cryptographic hedge" motivated NIST's standardization, despite the continued security of SHA-2. Further, ref. [32] highlighted the historical pattern of blockchain vulnerability attacks on Merkle–Damgård designs (MD5, SHA-0, SHA-1), necessitating alternative constructions for long-term security.

Length Extension Attack Immunity: SHA-256, like all SHA-2 algorithms, is vulnerable to length extension attacks where an attacker, knowing H(m), can compute H(m ||

m′) for chosen m′ without knowing m [33]. This vulnerability requires careful implementation, typically using HMAC constructions. SHA-3's sponge construction is inherently immune to length extension attacks [30], simplifying secure implementation and reducing the risk of developer error, a critical consideration given smart contract vulnerabilities [34].

Equivalent Security Strength: Both SHA-256 and Keccak-256 provide 128-bit collision resistance and 256-bit preimage resistance, offering equivalent security for blockchain applications [18,30]. The collision resistance of 2^128 operations exceeds the feasibility of practical attacks, even with anticipated technological advances. [35]. This equivalence means that the choice between algorithms rests on other factors, including architectural diversity of algorithms used in various blockchains, attack immunity, and ecosystem compatibility.

Performance Trade-offs: On traditional Central Processing Unit (CPU) architectures, SHA-256 generally outperforms Keccak-256. Benchmarks show SHA-256 at approximately 5.6 cycles per byte versus Keccak-256 at 11.7–12.25 cycles per byte on x86-64 processors [30]. However, in hardware implementations (Application-Specific Integrated Circuits (ASICs), Field-Programmable Gate Array (FPGAs), SHA-3 demonstrates superior performance due to its permutation-based design, amenable to parallel processing [30]. A recent study by [36] regarding AI-driven blockchain optimization posited that as blockchain infrastructure increasingly employs specialized hardware accelerators, the architectural properties of SHA-3 become advantageous.

Blockchain Adoption and Rationale: While Bitcoin employs SHA-256, Ethereum made a deliberate choice to use Keccak-256 for several reasons [12,37]: (1) ASIC resistance during the design phase, as Keccak's properties initially deterred specialized mining hardware; (2) flexibility in output length through Secure Hash Algorithm-Keccak (SHAKE) variants, enabling domain-specific hash sizes; (3) clean mathematical foundations providing confidence in long-term security. Furthermore, ref. [38] examined the challenge of identifying vulnerabilities in smart contracts, which is crucial for preventing criminals from executing malicious exploits during DeFi transactions. A case study by [37], regarding Ethereum Classic's security responses to 51% attacks, recommended that switching to the Keccak-256 PoW algorithm, citing its NIST standardization, enhanced security as the latest SHA family member, and faster performance compared to Ethash.

Keccak vs. SHA-3 Clarification: A crucial technical distinction exists between original Keccak and standardized SHA-3. NIST modified Keccak's padding scheme when standardizing SHA-3, changing the padding constant from 0 × 01 to 0 × 06 [39]. This alteration produces different hash outputs despite identical core algorithms and security levels. Ethereum uses the original Keccak-256, rather than NIST's SHA-3-256, which is a common source of confusion when interfacing with other systems [37,39].

2.4. Byzantine Fault Tolerance (BFT)

BFT has become a foundational concept in distributed computing and blockchain consensus research, designed to enable reliable agreement among network participants even when some nodes behave unpredictably or maliciously. Traditional fault-tolerant systems primarily handle simple crash failures, but BFT extends resilience to cover arbitrary and deceptive behaviours, as originally framed by the Byzantine Generals Problem. The central objective of BFT mechanisms is to ensure that honest nodes can still achieve agreement on the correct state of the system, provided that no more than one-third of participants are compromised. Early theoretical research into BFT laid the groundwork for dependable distributed systems, although these early models faced scalability and communication inefficiencies. The introduction of the Practical Byzantine Fault Tolerance (PBFT) algorithm marked a significant step forward by enabling consensus in asynchronous environments with lower latency. However, as noted in [16], these early BFT solutions were primarily developed for closed and permissioned networks, making them unsuitable for decentralized public systems, such as blockchains.

Recent advancements in blockchain technology have redefined the application of BFT principles. Ref. [2] emphasized that modern consensus protocols integrate BFT mechanisms with PoS and hybrid architectures to enhance efficiency, scalability, and energy conservation. These newer designs, such as Tendermint and HotStuff, achieve deterministic finality, meaning that once a block is validated, it becomes irreversible, unlike the probabilistic nature of PoW systems. Similarly, ref. [19] explored scalability limitations in decentralized networks, highlighting that while BFT provides robust security, optimizing communication complexity remains critical to achieving higher throughput and real-time performance. Overall, the evolution of BFT from theoretical constructs to practical blockchain consensus mechanisms reflects the growing demand for decentralized yet reliable systems. Modern implementations strike a balance between security and scalability, underpinning a new generation of blockchain protocols that achieve distributed trust without relying on centralized authorities.

2.5. Contemporary Blockchain Vulnerabilities

2.5.1. Maximal/Blockchain Extractable Value (MEV/BEV) Attacks

MEV, introduced by [4] in their seminal "Flash Boys 2.0" research, represents a category of attacks where blockchain actors, originally miners, now validators and searchers, manipulate transaction ordering to extract profit. The concept has since been generalized to BEV, which encompasses all forms of value extraction through transaction ordering, regardless of the actor's role in the consensus mechanism [16].

Attack Mechanisms and Financial Impact: ref. [12]'s comprehensive quantification of BEV reveals the magnitude of this threat, over 32 months (analyzed period ending 2022), BEV extraction generated USD 540.54 million in profit across 11,289 unique addresses,

spanning 49,691 cryptocurrencies and 60,830 on-chain markets. Subsequent data indicate continued growth, with over USD 500 million extracted through September 2022 alone, and more than 500,000 ETH (exceeding USD 1 billion) extracted since that date. Recent analysis by [40] confirms ongoing BEV activity, with sandwich attacks, liquidations, and transaction replay collectively generating significant losses. Three primary BEV attack strategies have been identified and quantified to include sandwich attacks, liquidation MEV bot attack, and desk arbitrage [4,16].

Sandwich Attacks (51.56% of total BEV, USD 289.76 million): The most prevalent BEV mechanism operates as follows: (1) An attacker monitors the public mempool for pending large-value trades. (2) The attacker submits a "frontrun" transaction purchasing the same asset immediately before the victim's transaction, driving the price upward. (3) The victim's transaction executes at an artificially inflated price. (4) The attacker submits a "backrun" transaction selling the asset at a profit. This three-phase attack exploits the predictable price impact of large trades combined with the attacker's ability to control transaction ordering through gas price manipulation [16] as depicted in Figure 1.

Liquidation MEV: DeFi lending protocols (Compound, Aave, MakerDAO) require borrowers to maintain collateral ratios above specified thresholds. When collateral value drops below these thresholds, often during market volatility, positions become liquidatable. MEV extractors deploy monitoring bots to detect liquidatable positions and race to submit liquidation transactions, earning protocol-defined bonuses (typically 5–10% of the liquidated position value). The competitive nature of liquidation MEV often results in Priority Gas Auctions (PGAs), where multiple actors bid increasingly high gas prices to ensure their transactions are included in the priority queue [16].

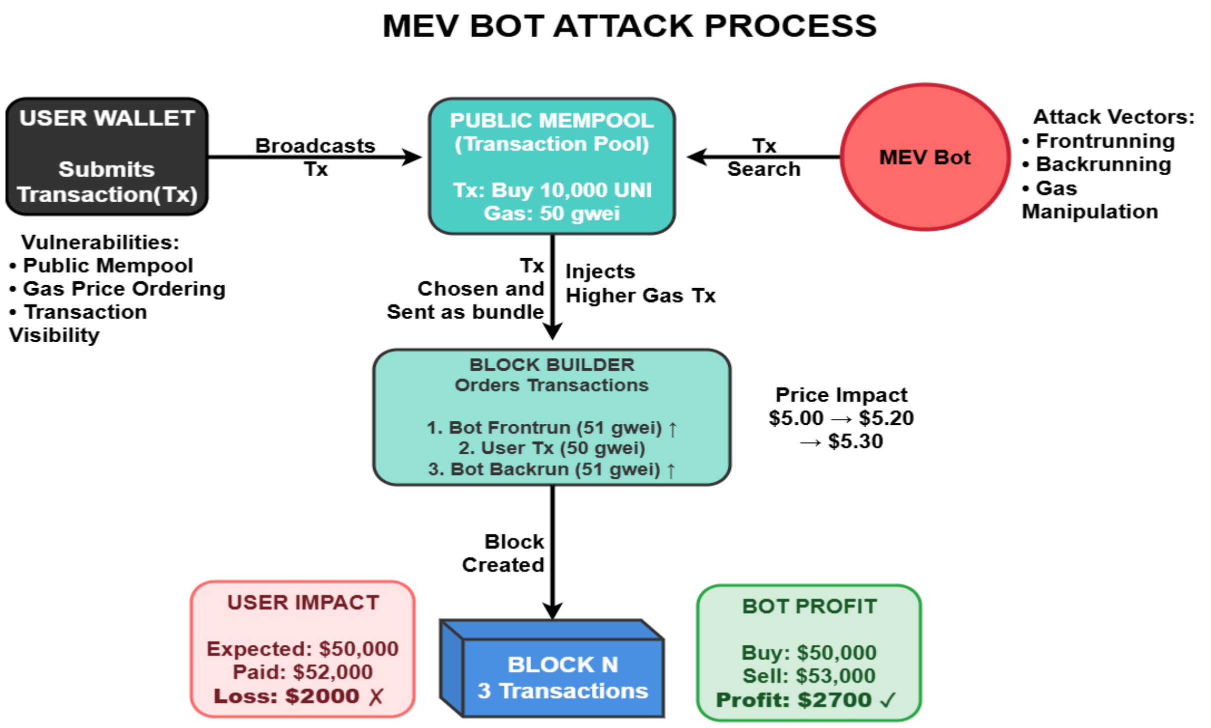


Figure 1. MEV/BEV bot attack process.

Decentralised Exchange (DEX) Arbitrage: Price discrepancies between decentralized exchanges create arbitrage opportunities that MEV Bots exploit. Traditional arbitrage requires upfront capital, but flash loans, uncollateralized loans that must be repaid within the same transaction, enable capital-free arbitrage. While arbitrage theoretically improves market efficiency, MEV-driven arbitrage often captures value that would otherwise benefit regular traders, and the competitive gas auctions for arbitrage opportunities contribute to network congestion [16].

Consensus Security Implications:

Beyond financial losses to individual users, BEV poses fundamental threats to blockchain consensus security [4,16]. The economic model is stark when BEV opportunities exceed block rewards; rational miners/validators have financial incentives to fork the blockchain to capture that value. Ref. [16] quantified this risk, demonstrating that:

- A miner with 10% of the network hash rate will rationally fork Ethereum if the BEV opportunity exceeds four times the standard block reward
- The highest observed BEV instance reached USD 4.1 million USD, equivalent to 616.6 times the Ethereum block reward at that time
- Such opportunities create time-bandit attack scenarios where future miners are incentivized to rewrite blockchain history if sufficient BEV exists in past blocks [4]. This economic threat model directly challenges security assumptions underlying PoW and PoS consensus mechanisms. While traditional blockchain security analysis focuses on the computational or financial cost of gaining majority control (>50% hashrate or stake), BEV attacks succeed with far lower thresholds when block rewards are exceeded by extractable value [4,16].

2.5.2. MEV/BEV Attacks and CIA Triad Violations

MEV/BEV attacks violate all three CIA triad principles in distinct ways. For instance, **Confidentiality:** Transaction privacy is fundamentally compromised by BEV mechanics. Users submit transactions to the public mempool, where they remain visible, including sender, receiver, amount, and smart contract calls, until miners/validators include them in blocks. This transparency, while necessary for network consensus, enables attackers to analyze pending transactions and identify profitable opportunities for frontrunning [4]. Even with strong cryptographic signatures (ECDSA), the transaction content itself remains vulnerable to confidentiality breaches, exposing users to targeted attacks. Sophisticated MEV searchers employ mempool monitoring tools that analyze thousands of pending transactions per second, identifying and exploiting vulnerable trades within milliseconds [industry analysis]. **Integrity:** Blockchain integrity traditionally refers to immutability, the property that historical transactions cannot be altered. However, BEV attacks introduce a distinct integrity threat: the manipulation of transaction ordering within blocks [4,16]. While block contents remain cryptographically secured (via SHA-3 or SHA-256 hashing), the sequence of transactions within blocks is subject to manipulation by block producers. When miners/validators can reorder, insert, or censor transactions to maximize extracted value, the blockchain's promise of fair, deterministic execution is violated. Furthermore, the economic incentive to fork the chain for ample BEV opportunities threatens finality, which is the assurance that once transactions are confirmed, they will not be reversed [16]. **Availability:** Fair access to blockchain networks is compromised by MEV/BEV competition [4,16]. Priority gas auctions, where MEV Bots bid increasingly high gas prices to ensure transaction inclusion ahead of competitors, create two availability problems:

(1) Network congestion: Competitive bidding during high-value MEV opportunities floods the network with transactions, causing delays for regular users.

(2) Economic exclusion: Users unable or unwilling to pay premium gas prices experience transaction delays or failures. Analysis shows that during periods of intense MEV competition, median gas prices can spike by 10–100×, effectively pricing out regular users [16]. Additionally, "failed transaction" attacks, where MEV bots intentionally submit transactions that fail after observing mempool conditions, waste network resources while users still incur gas fees for the failed execution.

2.5.3. Cross-Chain Bridge Exploits and Vulnerability Categories

As blockchain ecosystems proliferate, cross-chain bridges that enable asset transfers between blockchains have become critical infrastructure and significant vulnerability vectors. Chainalysis reports bridges account for USD 2.8 billion in losses, representing 40% of all cryptocurrency hacks [3]. The following explains the vulnerability categories.

Smart Contract Bugs: The Poly Network hack (2021) exploited a privilege escalation vulnerability in the bridge's contract logic, enabling the attacker to forge cross-chain messages and withdraw USD 611 million [41]. The Wormhole bridge hack (2022) similarly stemmed from a signature verification bypass, resulting in USD 325 million losses [42].

Consensus Manipulation: Bridges relying on external validators are vulnerable to risks from validator compromise or collusion. The Ronin Bridge hack (2022) was successful when attackers compromised five of the nine validator keys, allowing for an unauthorized withdrawal of USD 625 million [43].

Wrapped Asset Vulnerabilities: Bridges that create wrapped representations of assets on destination chains introduce risks related to peg maintenance and reserve management. If the bridge's reserves on the source chain are compromised, wrapped assets become unbacked, which could potentially trigger cascading liquidations. Ref. [22] identifies cross-chain bridge vulnerabilities as a critical contemporary threat; bridge attacks target integration points between separate systems where security assumptions may not align. Figure 2 shows the cross-chain bridge attack mechanisms.

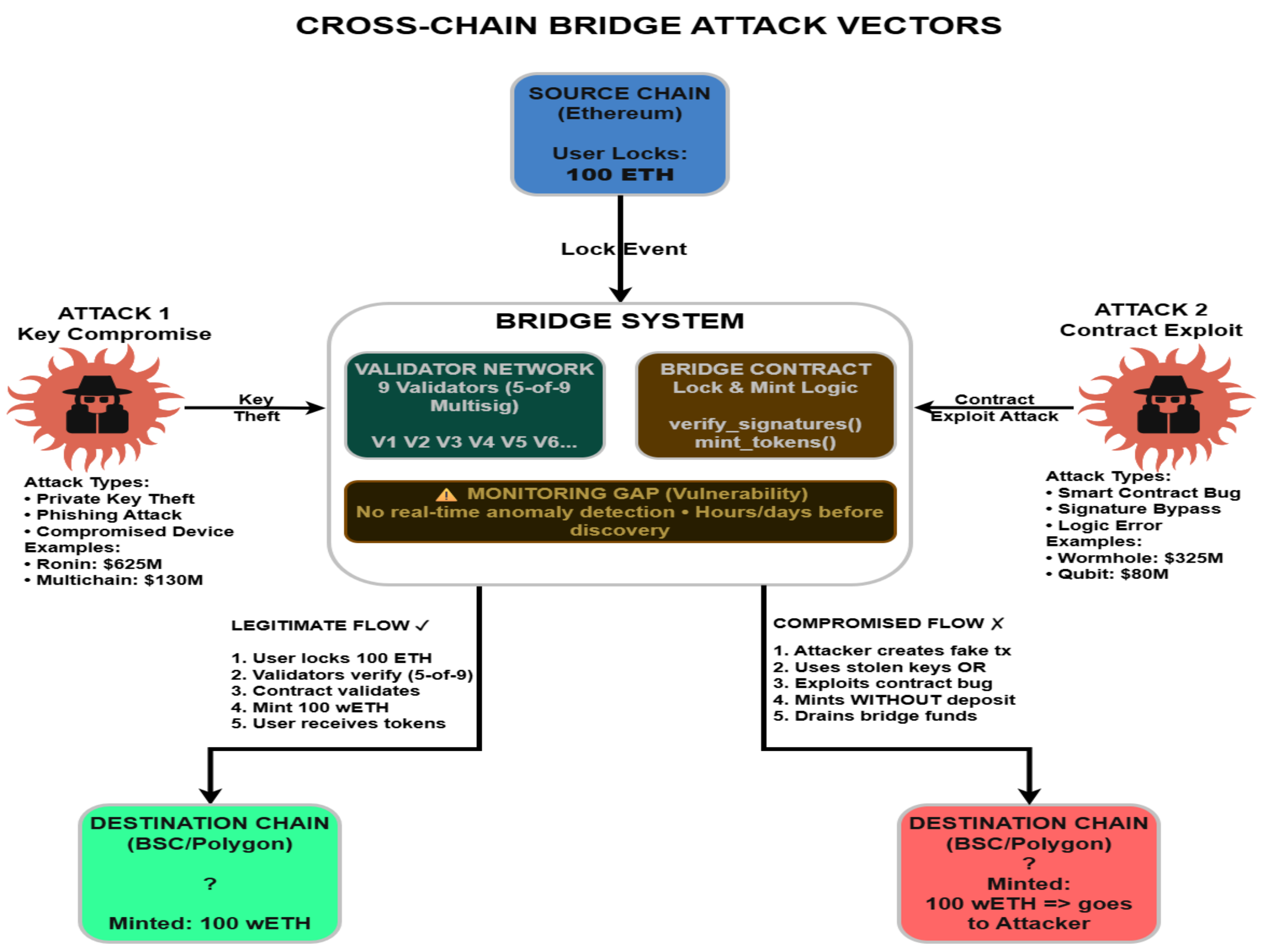


Figure 2. Cross-chain bridge attack process.

2.5.4. Consensus Layer Vulnerabilities

Blockchain consensus mechanisms, while robust against classical attacks, face evolving threats like 51% attacks, long-range attacks, and selfish mining.

51% Attacks: PoW blockchains with low hash rates remain vulnerable to majority attacks, where adversaries controlling more than 50% of the hash power can double-spend and censor transactions. Ethereum Classic suffered repeated 51% attacks in 2020, resulting in millions of dollars lost [44]. The analysis of [32] notes that to date, no majority PoW network that dominates its hardware class has been successfully affected by 51% attacks, highlighting the importance of hash rate distribution. Both PoW and PoS attack mechanisms are illustrated in Figure 3 threat model process.

Long-Range Attacks: PoS systems face unique "nothing-at-stake" problems, where validators can validate multiple conflicting forks at a lower cost. Long-range attacks exploit this by constructing alternative histories from the genesis [12].

Selfish Mining: Miners withhold discovered blocks to gain unfair rewards, reducing network efficiency and fairness [17]. This attack succeeds with as little as 25% hash power under certain network conditions [45] provides a blockchain vulnerability taxonomy that categorizes attacks across applications, smart contracts, consensus, network, data, and infrastructure layers during integration points.

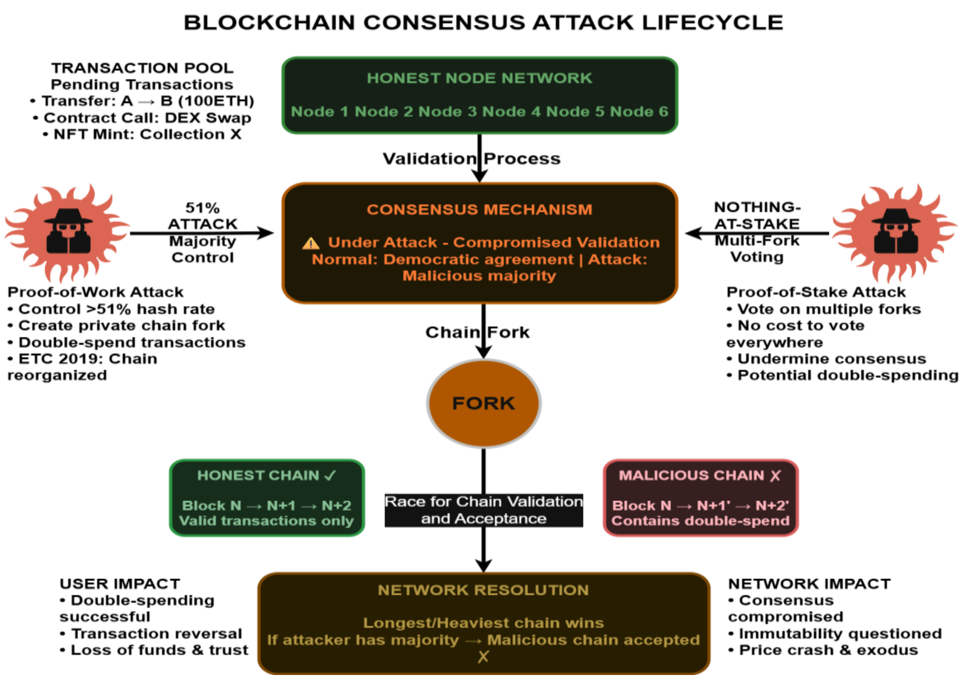

BLOCKCHAIN CONSENSUS ATTACK LIFECYCLE
TRANSACTION POOL
Pending Transactions
• Transfer: A → B (100ETH)
• Contract Call: DEX Swap
• NFT Mint: Collection X
HONEST NODE NETWORK
Node 1 Node 2 Node 3 Node 4 Node 5 Node 6
Validation Process
51% ATTACK Majority Control
CONSENSUS MECHANISM
⚠ Under Attack - Compromised Validation
Normal: Democratic agreement | Attack: Malicious majority
NOTHING-AT-STAKE Multi-Fork Voting
Proof-of-Work Attack
• Control >51% hash rate
• Create private chain fork
• Double-spend transactions
• ETC 2019: Chain reorganized
Chain Fork
Proof-of-Stake Attack
• Vote on multiple forks
• No cost to vote everywhere
• Undermine consensus
• Potential double-spending
FORK
HONEST CHAIN ✓
Block N → N+1 → N+2
Valid transactions only
Race for Chain Validation and Acceptance
MALICIOUS CHAIN ✗
Block N → N+1' → N+2'
Contains double-spend
USER IMPACT
• Double-spending successful
• Transaction reversal
• Loss of funds & trust
NETWORK RESOLUTION
Longest/Heaviest chain wins
If attacker has majority → Malicious chain accepted ✗
NETWORK IMPACT
• Consensus compromised
• Immutability questioned
• Price crash & exodus

Figure 3. Blockchain consensus layer threat model process.

2.6. Research Gap Identification

While existing literature comprehensively addresses several aspects of blockchain security, several gaps remain due to technological changes, a changing attack surface, and evolving threats.

Cryptographic Transition Guidance: Limited practical guidance exists for migrating blockchain implementations from legacy cryptographic primitives (RSA, SHA-256) to modern alternatives (ECDSA, SHA-3), particularly regarding performance implications and compatibility management.

Integrated CIA Framework: Most research addresses confidentiality, integrity, or availability in isolation. Comprehensive frameworks that demonstrate practical implementation of the CIA triad with modern cryptography in working blockchain systems remain scarce.

Contemporary Threat Integration: Recent vulnerabilities (MEV/BEV, 51% Attacks) are well-documented individually, but their integration into comprehensive blockchain security frameworks and mapping to CIA triad principles requires further investigation.

Practical Implementation Details: Academic blockchain research often remains theoretical or uses simplified implementations. Production-grade code examples demonstrated security patterns are limited, creating knowledge gaps for practitioners.

Hybrid Consensus Mechanism: This system is designed to utilize an alternating PoW and PoS mechanism for block generation, alternating between odd and even numbers in a sequence. It then uses BFT voting for block validation and confirmation. This three-level hybrid consensus mechanism is fully demonstrated to provide a comprehensive understanding of consensus mechanisms, a rarity in contemporary research.

This research addresses these gaps by providing MasterChain, a practical blockchain implementation featuring modern cryptographic primitives, comprehensive vulnerability analysis, and detailed implementation guidance for the CIA triad principles, which incorporates a hybrid consensus mechanism.

Limitations in Previous Works: Existing blockchain security research frequently references the CIA triad as a foundational security model. However, most studies apply it conceptually rather than operationally. Further, ref. [5] employ the CIA triad as an evaluative lens to assess blockchain adoption in public sector applications, extending the model with non-repudiation. While insightful at a governance level, their work does not address the implementation of CIA properties within blockchain communication mechanisms. Furthermore, ref. [6] provides a comprehensive survey of blockchain applications in CPS and IoT systems, highlighting how CIA can theoretically be maintained in distributed communication environments. However, the study remains architectural CPS and IoT systems and does not present an executable communication model or empirical validation. Additionally, [46] propose a blockchain-based framework for secure IoT communication using smart contracts. Although the work addresses integrity and availability through permissioned blockchain infrastructure, confidentiality is largely enforced through access control assumptions rather than cryptographic protection of communication payloads. Moreover, ref. [47] integrate the CIA triad with STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege) threat modelling to secure peer-to-peer networks via blockchain-based access control. Their focus is on authorization and policy enforcement rather than end-to-end communication security. However, this study implements the CIA triad directly within blockchain communication, demonstrating how CIA can be enforced and evaluated during transaction exchange and node interaction.

Recent blockchain security research [7] has emphasized cryptographic, privacy-preserving, and trust-enhancing mechanisms across multiple application domains. While such studies provide valuable theoretical and methodological insights, they typically address security primitives in isolation. However, our work focused on the system-level enforcement of the CIA triad within blockchain communication. The proposed MasterChain implementation integrates CIA directly into transaction propagation and node interaction, rather than treating them as independent design considerations. Thus, our study contributes to the improvements made by [7] and demonstrates how established security principles can be operationalized cohesively to secure blockchain communication, extending existing work from mechanism-centric analysis to communication-focused implementation. Section 4.4 details the practical realization of these properties within the MasterChain testbed, addressing gaps between theoretical security models and real blockchain communication systems.

3. Approach

This section discusses the approach and how we employed the design science methodology, drawing on Hevner's design science framework [48]. The approach combines constructive research through artifact development (MasterChain implementation) with empirical evaluation of security properties and performance characteristics. Further, the framework emphasizes creating innovative artifacts that address identified problems while contributing to the knowledge base through rigorous evaluation and assessment. Thus, the approach is relevant for our implementation.

Performance Reality in Practice: While CPU benchmarks show SHA-256 approximately 2x faster than Keccak-256 [30]. The actual impact on blockchain operations is minimal. Our testing (Section 5) reveals that for a typical block (100 transactions, ~15 KB), Keccak-256 requires 0.11 ms, versus SHA-256's theoretical 0.05 ms—a 0.06 ms difference that is negligible compared to the 12-s block times. Our methodology comprises four interconnected phases:

Phase 1: Requirements Analysis—Systematically identifying security requirements through the CIA triad decomposition and analysis of the contemporary vulnerability landscape. This phase established functional and non-functional requirements for MasterChain.

Phase 2: Cryptographic Selection and Justification—Comparative evaluation of cryptographic primitives (ECDSA vs. RSA, SHA-3 vs. SHA-256) based on security properties, performance metrics, blockchain compatibility, and future-readiness criteria.

Phase 3: Design and Implementation—Development of MasterChain architecture incorporating selected cryptographic primitives, distributed node communication, and consensus mechanisms. The implementation utilized a Python tool, with an emphasis on code clarity and extensibility, enhanced by an intuitive web interface and automated networking features.

Phase 4: Evaluation and Validation—Empirical testing of CIA triad properties, performance benchmarking, and security analysis against known vulnerability classes.

4. Implementation

This section discusses the implementation of blockchain security mechanisms using the MasterChain tool in a Virtual Machine (VM) environment as a testbed to demonstrate how cryptography and hashing algorithms are used in a distributed network to enhance CIA and security.

4.1. CIA Triad Security Implementation

Recent CIA triad in blockchain security research has emphasized issues of confidentiality and trust through advanced cryptographic and privacy-preserving techniques, such as encryption frameworks and zero-knowledge proofs [5,7]. However, such approaches are typically presented as standalone mechanisms or design options rather than as components of a unified communication security model [6,46]. In contrast, the implementation presented in this work integrates CIA simultaneously within the blockchain communication workflow. Specifically, confidentiality in MasterChain is enforced at the message-exchange level, ensuring that transaction payloads remain protected during node-to-node transmission rather than relying solely on ledger-level access control [46]. Integrity is actively verified during communication and block propagation, extending beyond post-consensus immutability guarantees commonly emphasized in conceptual assessments [5]. Availability is treated as a measurable system property, supported through decentralized node replication and evaluated under communication disruption scenarios, aligning with security expectations identified in distributed and peer-to-peer systems [6,47].

This integrated approach differentiates the proposed implementation from existing blockchain security studies that focus on isolated security primitives [5,7]. By embedding CIA requirements directly into communication processes, the proposed system demonstrates how classical security principles can be operationalized within blockchain environments. Unlike prior blockchain security approaches that emphasize ledger immutability or access control alone [7], this implementation enforces CIA as interdependent communication-level properties, enabling the CIA triad to function as a practical design framework for secure blockchain communication rather than a post-hoc evaluative model.

4.2. MasterChain Architecture

The MasterChain tool is used to implement a distributed blockchain system comprising three primary components: the blockchain core, consensus mechanism, and network layer.

1. **Blockchain Core:** Data structures representing blocks, transactions, and the chain itself with ECDSA signatures and Keccak-256 hashing algorithms.
2. **Consensus Mechanism:** A hybrid system comprising PoW and PoS implementation requiring the use of either computational effort or staked amount to generate and

propose a block using the Keccak-256 hashing algorithm, and also validate the proposed block. BFT algorithm for voting on proposed blocks for confirmation and adoption into the chain.

3. **Network Layer:** Flask-based Representational State Transfer (REST) Application Programming Interface (API) enabling peer discovery through automated node communication and distributed consensus

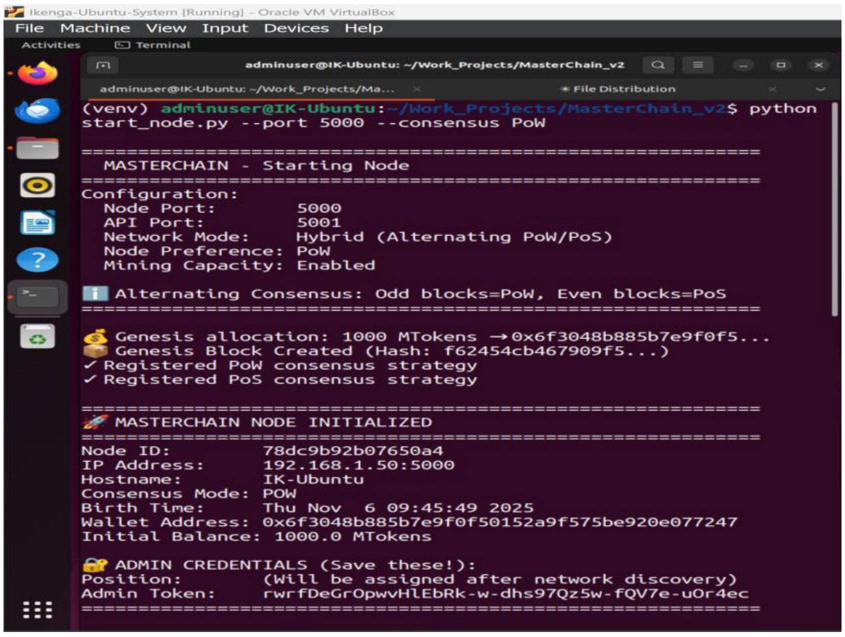
Figure 4 illustrates the high-level architecture.

```
┌──────────────────────────────────────────────────
│                  MasterChain Node                        │
├──────────────────────────────────────────────────
│  Flask REST API Layer                                     │
│  ├─ /transactions/new      (Submit transactions)          │
│  ├─ /mine                  (Trigger block mining)         │
│  ├─ /chain                 (Retrieve blockchain)          │
│  ├─ /nodes/register        (Register peer nodes)          │
│  └─ /nodes/resolve         (Consensus resolution)         │
├─────────────────────────────────────────────────┤
│  Blockchain Core                                          │
│  ├─ Transaction Pool (Unconfirmed transactions)           │
│  ├─ Block Construction (Bundle transactions into blocks)  │
│  ├─ Chain Management (Maintain block sequence)            │
│  └─ Consensus Validation (Verify PoW/PoS, Peer Voting)    │
├─────────────────────────────────────────────────┤
│  Cryptographic Layer                                      │
│  ├─ ECDSA Operations (secp256k1 signing/verification)     │
│  ├─ Key Management (Private key protection)               │
│  ├─ Keccak-256 Hashing (Block and transaction hashing)    │
│  └─ PoW (Hash-based computational puzzle)                 │
│  └─ PoS (Hash-based generation )                          │
├─────────────────────────────────────────────────┤
│  Network Communication                                    │
│  ├─ Automated Peer Discovery (Register neighboring nodes)  │
│  ├─ Chain Synchronization (Via chain consensus)           │
│  └─ Transaction Broadcasting (Propagate to network)       │
│  └─ BFT Voting System (Block confirmation and adoption)   │
└──────────────────────────────────────────────────────────┘
```

Figure 4. MasterChain node architecture showing layered design from network communication through cryptographic operations to the REST API interface.

4.3. MasterChain Set up and Implementation

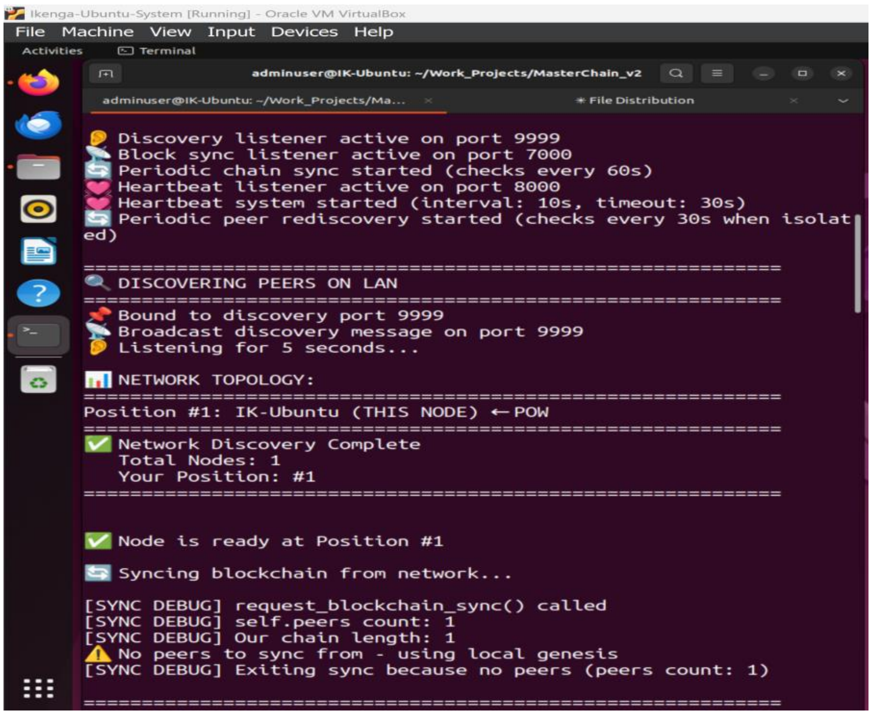
The Lab setup for the implementation process is fully detailed in the supplementary materials included. This involves creating a virtual environment and installing the requirements. The following command is used in a Linux command line to activate the Python virtual environment: *source name_of_virtual/bin/activate,* where the variable "name_of_virtual" is the name used to create the virtual environment. The following command is used to start up the MasterChain system: *python start_node.py -- port 5000 -- consensus PoW* (this starts the Masterchain system in PoW mode) or *python start_node.py -- port 5000 -- consensus PoS -- stake 100* (for starting the Masterchain system in PoS mode with a Stake of 100). Figure 5 illustrates the console output of the MasterChain system when it is initiated, utilizing a PoW consensus mode for block mining, and running at API port 5001 for operations interface communications. It also shows the registration of available consensus strategies and wallet addresses for transactions.


Ikenga-Ubuntu-System [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
adminuser@IK-Ubuntu: ~/Work_Projects/MasterChain_v2
adminuser@IK-Ubuntu: ~/Work_Projects/Ma...
File Distribution

```
(venv) adminuser@IK-Ubuntu:~/Work_Projects/MasterChain_v2$ python
start_node.py --port 5000 --consensus PoW

============================================================
  MASTERCHAIN - Starting Node
============================================================
Configuration:
  Node Port:        5000
  API Port:         5001
  Network Mode:     Hybrid (Alternating PoW/PoS)
  Node Preference:  PoW
  Mining Capacity:  Enabled

ℹ️ Alternating Consensus: Odd blocks=PoW, Even blocks=PoS
============================================================

💰 Genesis allocation: 1000 MTokens →0x6f3048b885b7e9f0f5...
📦 Genesis Block Created (Hash: f62454cb467909f5...)
✓ Registered PoW consensus strategy
✓ Registered PoS consensus strategy

============================================================
🚀 MASTERCHAIN NODE INITIALIZED
============================================================
Node ID:         78dc9b92b07650a4
IP Address:      192.168.1.50:5000
Hostname:        IK-Ubuntu
Consensus Mode:  POW
Birth Time:      Thu Nov  6 09:45:49 2025
Wallet Address:  0x6f3048b885b7e9f0f50152a9f575be920e077247
Initial Balance: 1000.0 MTokens

🔐 ADMIN CREDENTIALS (Save these!):
Position:        (Will be assigned after network discovery)
Admin Token:     rwrfDeGrOpwvHlEbRk-w-dhs97Qz5w-fQV7e-uOr4ec
============================================================
```

Figure 5. First node (IK-Ubuntu) showing the MasterChain start-up prompt.

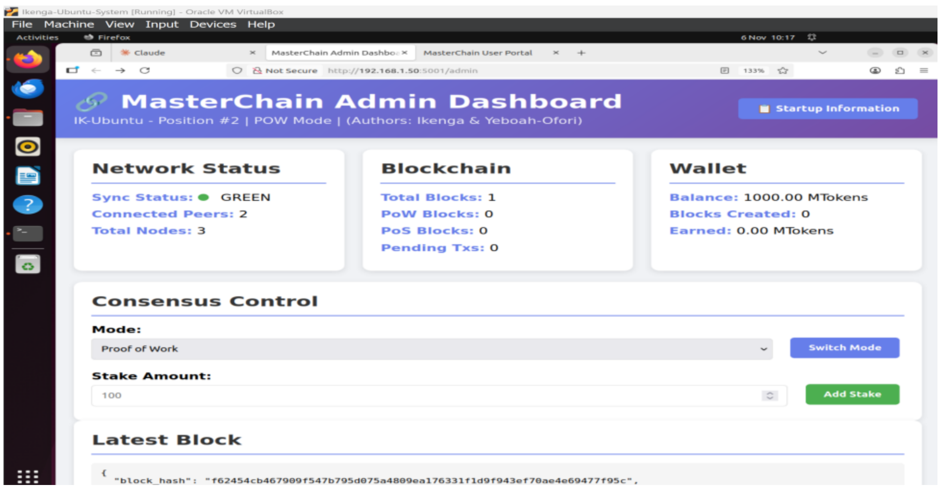
Figure 6 shows the network ports initialization sequence, network detection, and peer discovery. It activated ports 9999, 7000, and 8000. It also checked for any already available nodes running, but found none, which prompted it to assume the first node position, thereby creating the Genesis block.


Ikenga-Ubuntu-System [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
adminuser@IK-Ubuntu: ~/Work_Projects/MasterChain_v2
adminuser@IK-Ubuntu: ~/Work_Projects/Ma...
File Distribution

```
👂 Discovery listener active on port 9999
📡 Block sync listener active on port 7000
🔄 Periodic chain sync started (checks every 60s)
💓 Heartbeat listener active on port 8000
💓 Heartbeat system started (interval: 10s, timeout: 30s)
🔄 Periodic peer rediscovery started (checks every 30s when isolat
ed)

============================================================
🔍 DISCOVERING PEERS ON LAN
============================================================
📌 Bound to discovery port 9999
📡 Broadcast discovery message on port 9999
👂 Listening for 5 seconds...

📊 NETWORK TOPOLOGY:
============================================================
Position #1: IK-Ubuntu (THIS NODE) ←POW
============================================================
✅ Network Discovery Complete
   Total Nodes: 1
   Your Position: #1
============================================================


✅ Node is ready at Position #1

🔄 Syncing blockchain from network...

[SYNC DEBUG] request_blockchain_sync() called
[SYNC DEBUG] self.peers count: 1
[SYNC DEBUG] Our chain length: 1
⚠️ No peers to sync from - using local genesis
[SYNC DEBUG] Exiting sync because no peers (peers count: 1)

============================================================
```

Figure 6. Network discovery peers for detecting other nodes on the network.

The web interface for monitoring and controlling the system can be accessed at the automatically acquired IP and API port 5001. Figure 7 shows an Admin interface. Please refer to the supplementary materials for the detailed demonstration of the interface functionalities and usage.


Ikenga-Ubuntu-System [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Firefox 6 Nov 10:17
Claude
MasterChain Admin Dashbo
MasterChain User Portal
Not Secure http://192.168.1.50:5001/admin
133%
Startup Information
Switch Mode
Add Stake

MasterChain Admin Dashboard
IK-Ubuntu - Position #2 | POW Mode | (Authors: Ikenga & Yeboah-Ofori)

Network Status
Sync Status: ● GREEN
Connected Peers: 2
Total Nodes: 3

Blockchain
Total Blocks: 1
PoW Blocks: 0
PoS Blocks: 0
Pending Txs: 0

Wallet
Balance: 1000.00 MTokens
Blocks Created: 0
Earned: 0.00 MTokens

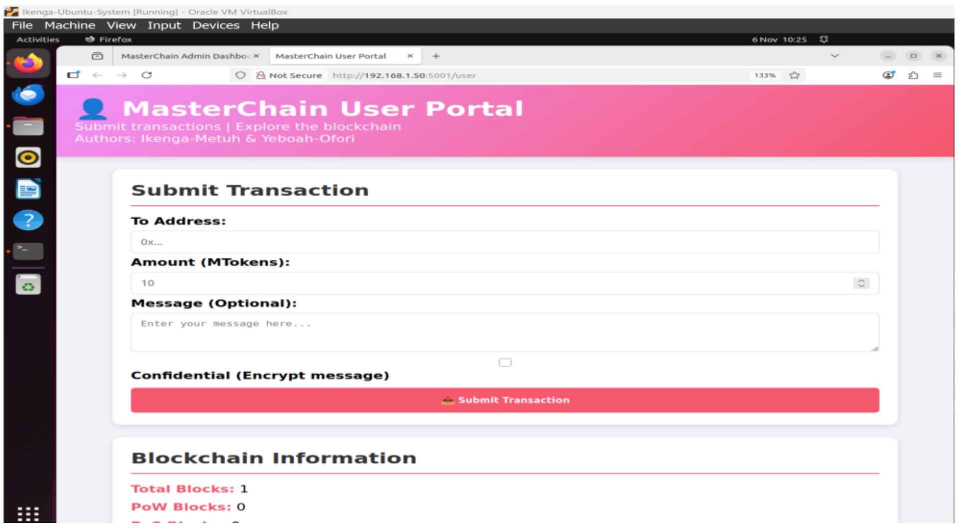
Consensus Control
Mode: Proof of Work
Stake Amount: 100

Latest Block

```
{
  "block_hash": "f62454cb467909f547b795d075a4809ea176331f1d9f943ef70ae4e69477f95c",
```

Figure 7. MasterChain admin dashboard for monitoring blockchain activities.

Figure 8 shows the User interface as it is up and running, displaying the MasterChain User Dashboard for Posting Transactions to the Blockchain.


Ikenga-Ubuntu-System [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Firefox 6 Nov 10:25
MasterChain Admin Dashbo
MasterChain User Portal
Not Secure http://192.168.1.50:5001/user
133%
Submit Transaction

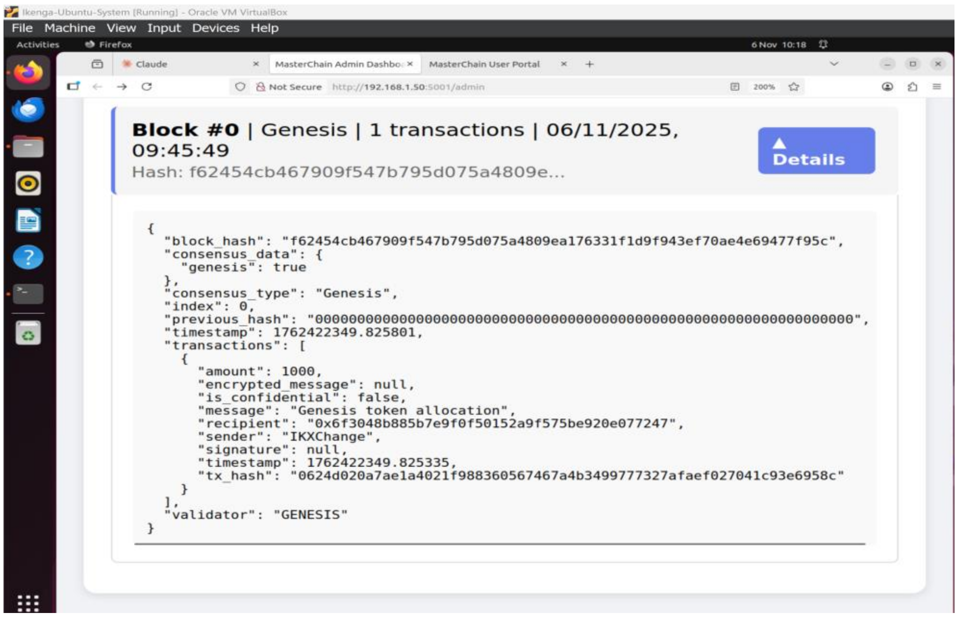
MasterChain User Portal
Submit transactions | Explore the blockchain
Authors: Ikenga-Metuh & Yeboah-Ofori

Submit Transaction
To Address: 0x...
Amount (MTokens): 10
Message (Optional): Enter your message here...
Confidential (Encrypt message)

Blockchain Information
Total Blocks: 1
PoW Blocks: 0

Figure 8. MasterChain user dashboard for posting transactions to the blockchain.

Figure 9 illustrates how the genesis block was initialized after it was automatically created. It also initializes with 1000 MTokens credited to the first running node, which can be sent to other nodes via cryptographic networking transaction communications. The genesis block is also generated with the previous hash as zeros, and its own block hash, which will serve as the foundation for the rest of the chain to build upon.

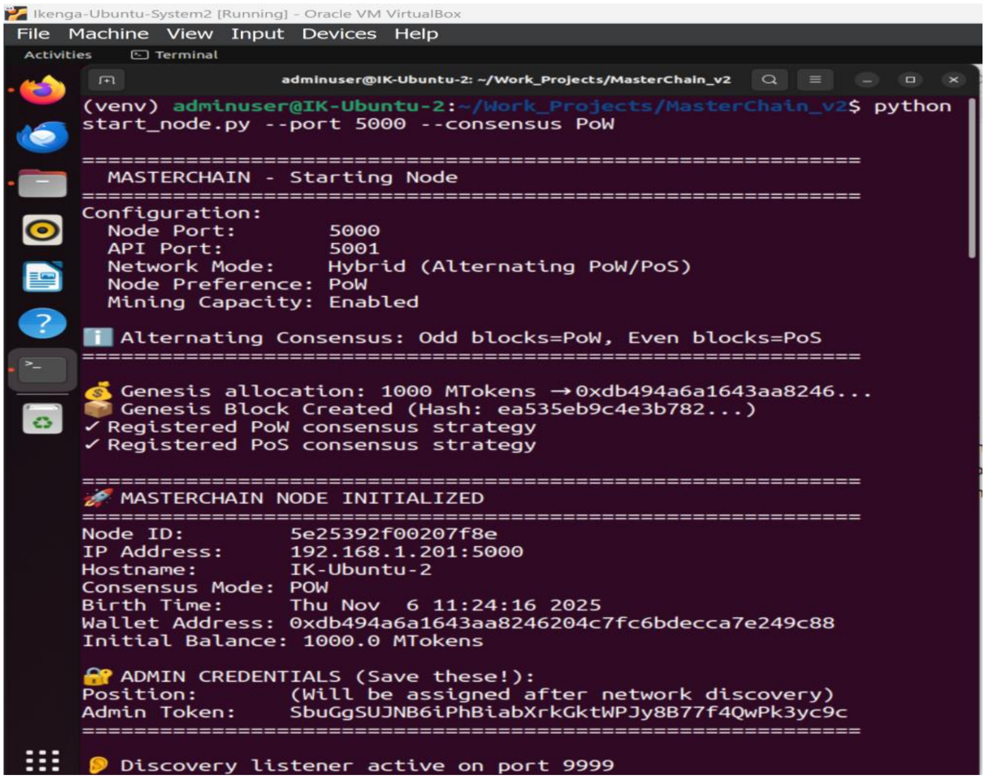

Ikenga-Ubuntu-System [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Firefox
6 Nov 10:18
Claude
MasterChain Admin Dashbo
MasterChain User Portal
Not Secure http://192.168.1.50:5001/admin
200%
Details

Block #0 | Genesis | 1 transactions | 06/11/2025, 09:45:49
Hash: f62454cb467909f547b795d075a4809e...

```
{
  "block_hash": "f62454cb467909f547b795d075a4809ea176331f1d9f943ef70ae4e69477f95c",
  "consensus_data": {
    "genesis": true
  },
  "consensus_type": "Genesis",
  "index": 0,
  "previous_hash": "0000000000000000000000000000000000000000000000000000000000000000",
  "timestamp": 1762422349.825801,
  "transactions": [
    {
      "amount": 1000,
      "encrypted_message": null,
      "is_confidential": false,
      "message": "Genesis token allocation",
      "recipient": "0x6f3048b885b7e9f0f50152a9f575be920e077247",
      "sender": "IKXChange",
      "signature": null,
      "timestamp": 1762422349.825335,
      "tx_hash": "0624d020a7ae1a4021f988360567467a4b3499777327afaef027041c93e6958c"
    }
  ],
  "validator": "GENESIS"
}
```

Figure 9. Genesis block generated for initializing blockchain transactions.

Figure 10 shows the startup of an additional node (IK-Ubuntu2). It also started with PoW mode as a block mining mechanism.

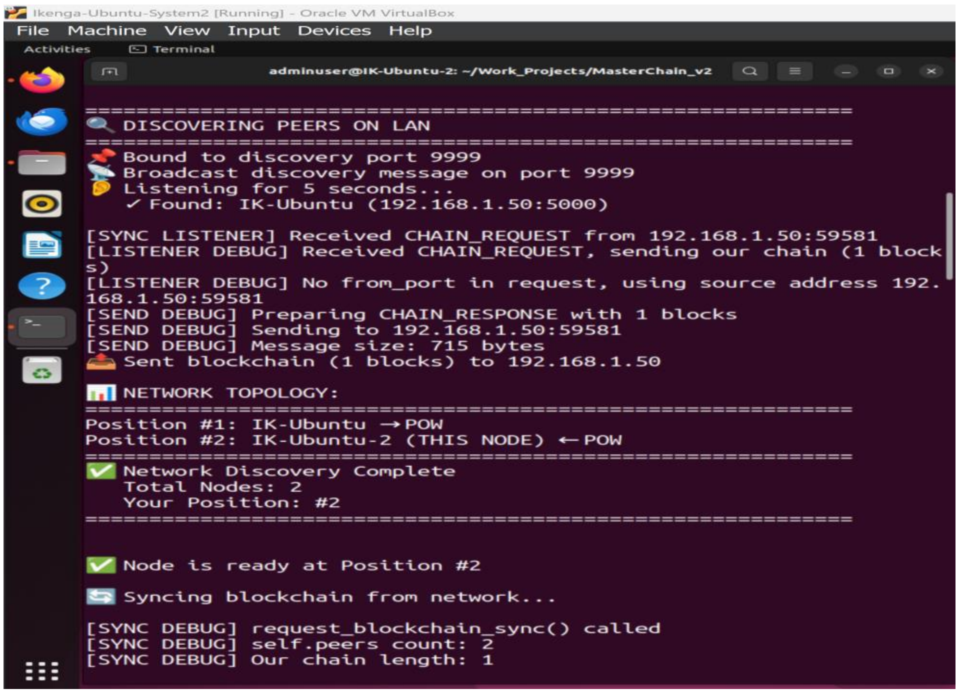

Ikenga-Ubuntu-System2 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
adminuser@IK-Ubuntu-2: ~/Work_Projects/MasterChain_v2

```
(venv) adminuser@IK-Ubuntu-2:~/Work_Projects/MasterChain_v2$ python
start_node.py --port 5000 --consensus PoW

============================================================
  MASTERCHAIN - Starting Node
============================================================
Configuration:
  Node Port:         5000
  API Port:          5001
  Network Mode:      Hybrid (Alternating PoW/PoS)
  Node Preference:   PoW
  Mining Capacity:   Enabled

ℹ️ Alternating Consensus: Odd blocks=PoW, Even blocks=PoS
============================================================

💰 Genesis allocation: 1000 MTokens →0xdb494a6a1643aa8246...
📦 Genesis Block Created (Hash: ea535eb9c4e3b782...)
✓ Registered PoW consensus strategy
✓ Registered PoS consensus strategy

============================================================
🚀 MASTERCHAIN NODE INITIALIZED
============================================================
Node ID:         5e25392f00207f8e
IP Address:      192.168.1.201:5000
Hostname:        IK-Ubuntu-2
Consensus Mode:  POW
Birth Time:      Thu Nov  6 11:24:16 2025
Wallet Address:  0xdb494a6a1643aa8246204c7fc6bdecca7e249c88
Initial Balance: 1000.0 MTokens

🔐 ADMIN CREDENTIALS (Save these!):
Position:        (Will be assigned after network discovery)
Admin Token:     SbuGgSUJNB6iPhBiabXrkGktWPJy8B77f4QwPk3yc9c
============================================================

📡 Discovery listener active on port 9999
```

Figure 10. IK-Ubuntu2 startup command for running the MasterChain system.

4.4. Multi-Node Deployment and Testing

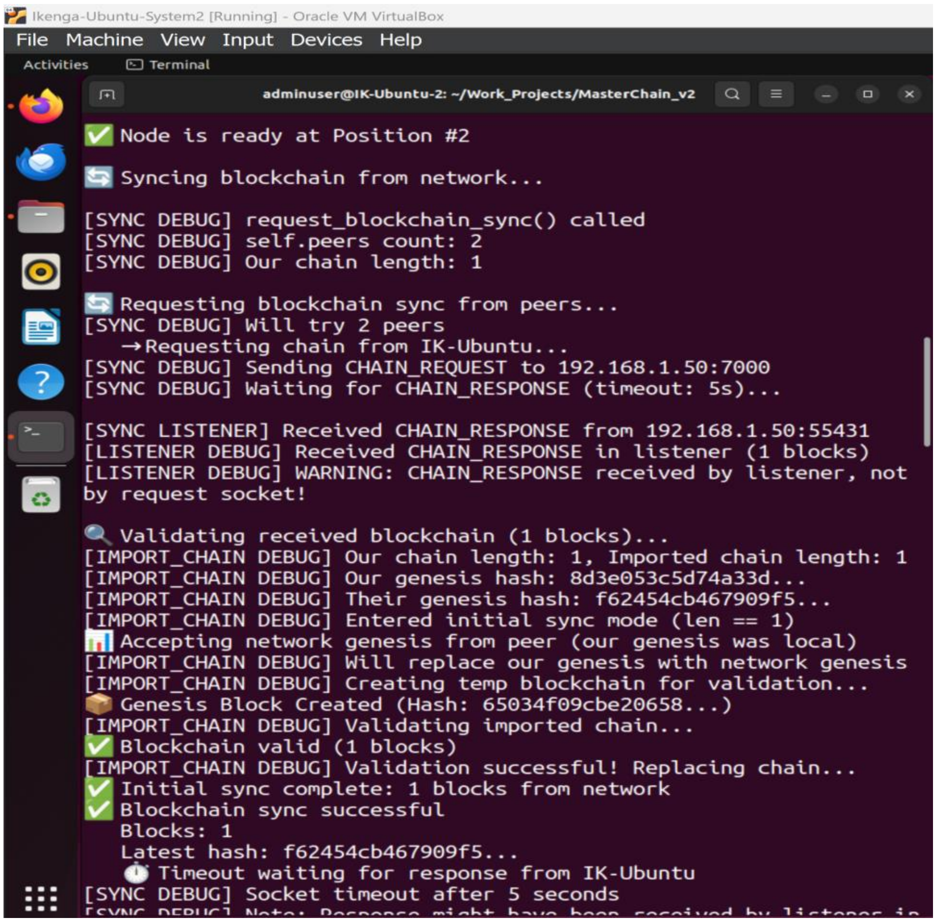
IK-Ubuntu2 starts up and enters discovery mode to automatically detect if a node already exists in the Local Area Network (LAN) running as a peer. It communicates with the discovered peer (IK-Ubuntu), assumes the next available position (number 2), and requests the blockchain, as seen in Figure 11.


Ikenga-Ubuntu-System2 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
adminuser@IK-Ubuntu-2: ~/Work_Projects/MasterChain_v2

```
============================================================
🔍 DISCOVERING PEERS ON LAN
============================================================
📌 Bound to discovery port 9999
📡 Broadcast discovery message on port 9999
👂 Listening for 5 seconds...
   ✓ Found: IK-Ubuntu (192.168.1.50:5000)

[SYNC LISTENER] Received CHAIN_REQUEST from 192.168.1.50:59581
[LISTENER DEBUG] Received CHAIN_REQUEST, sending our chain (1 block
s)
[LISTENER DEBUG] No from_port in request, using source address 192.
168.1.50:59581
[SEND DEBUG] Preparing CHAIN_RESPONSE with 1 blocks
[SEND DEBUG] Sending to 192.168.1.50:59581
[SEND DEBUG] Message size: 715 bytes
📤 Sent blockchain (1 blocks) to 192.168.1.50

📊 NETWORK TOPOLOGY:
============================================================
Position #1: IK-Ubuntu →POW
Position #2: IK-Ubuntu-2 (THIS NODE) ←POW
============================================================
✅ Network Discovery Complete
   Total Nodes: 2
   Your Position: #2
============================================================

✅ Node is ready at Position #2

🔄 Syncing blockchain from network...

[SYNC DEBUG] request_blockchain_sync() called
[SYNC DEBUG] self.peers count: 2
[SYNC DEBUG] Our chain length: 1
```

Figure 11. Network peer discovery and positioning for distributed communication.

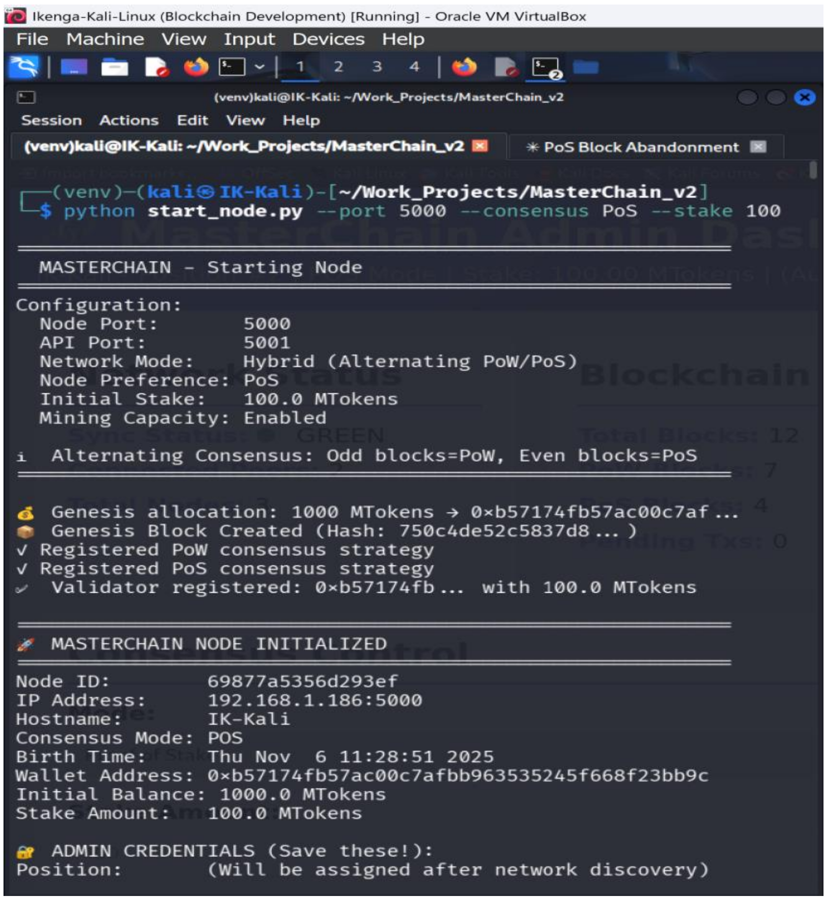
Figure 12 illustrates the process of network peer discovery and registration. IK-Ubuntu requests the existing chain and compares it with its locally created chain. With the help of timestamps, it detects that the network chain is older than its own newly created genesis block, which prompts it to accept the network chain (from IK-Ubuntu) and discard its own.


Ikenga-Ubuntu-System2 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
adminuser@IK-Ubuntu-2: ~/Work_Projects/MasterChain_v2

```
✅ Node is ready at Position #2

🔄 Syncing blockchain from network...

[SYNC DEBUG] request_blockchain_sync() called
[SYNC DEBUG] self.peers count: 2
[SYNC DEBUG] Our chain length: 1

🔄 Requesting blockchain sync from peers...
[SYNC DEBUG] Will try 2 peers
   →Requesting chain from IK-Ubuntu...
[SYNC DEBUG] Sending CHAIN_REQUEST to 192.168.1.50:7000
[SYNC DEBUG] Waiting for CHAIN_RESPONSE (timeout: 5s)...

[SYNC LISTENER] Received CHAIN_RESPONSE from 192.168.1.50:55431
[LISTENER DEBUG] Received CHAIN_RESPONSE in listener (1 blocks)
[LISTENER DEBUG] WARNING: CHAIN_RESPONSE received by listener, not
by request socket!

🔍 Validating received blockchain (1 blocks)...
[IMPORT_CHAIN DEBUG] Our chain length: 1, Imported chain length: 1
[IMPORT_CHAIN DEBUG] Our genesis hash: 8d3e053c5d74a33d...
[IMPORT_CHAIN DEBUG] Their genesis hash: f62454cb467909f5...
[IMPORT_CHAIN DEBUG] Entered initial sync mode (len == 1)
📊 Accepting network genesis from peer (our genesis was local)
[IMPORT_CHAIN DEBUG] Will replace our genesis with network genesis
[IMPORT_CHAIN DEBUG] Creating temp blockchain for validation...
📦 Genesis Block Created (Hash: 65034f09cbe20658...)
[IMPORT_CHAIN DEBUG] Validating imported chain...
✅ Blockchain valid (1 blocks)
[IMPORT_CHAIN DEBUG] Validation successful! Replacing chain...
✅ Initial sync complete: 1 blocks from network
✅ Blockchain sync successful
   Blocks: 1
   Latest hash: f62454cb467909f5...
   ⏱️ Timeout waiting for response from IK-Ubuntu
[SYNC DEBUG] Socket timeout after 5 seconds
[SYNC DEBUG] Note: Response might have been received by listener in
```

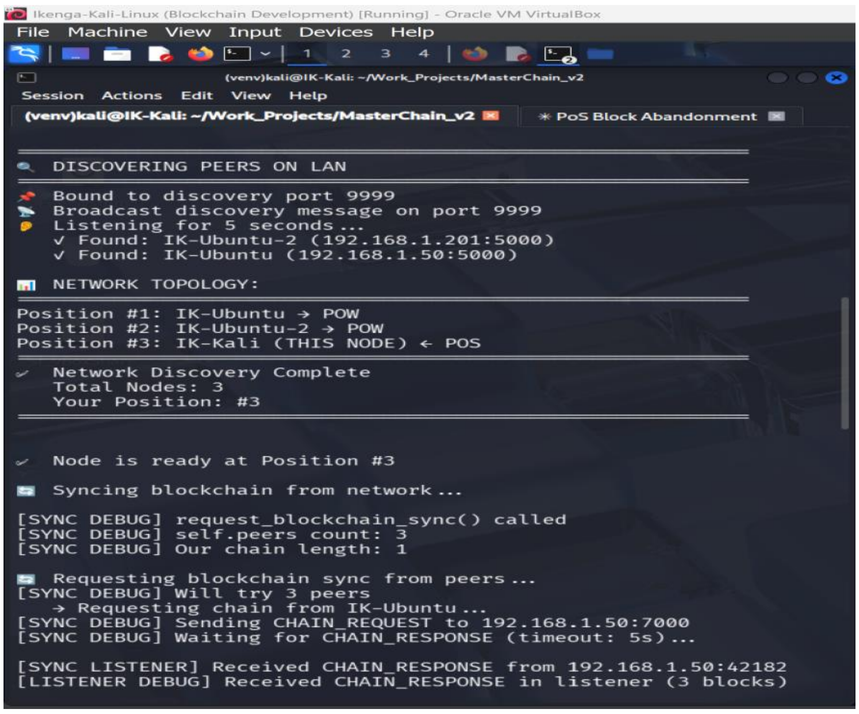
Figure 12. Blockchain request and synchronization between running nodes.

Figures 13 and 14 illustrate the startup processes for a third node (IK-Kali), specifically the discovery of network peers and registration. It began with a PoS block mining strategy option, featuring an initial stake of 100 MTokens.


Ikenga-Kali-Linux (Blockchain Development) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
(venv)kali@IK-Kali: ~/Work_Projects/MasterChain_v2
Session Actions Edit View Help
(venv)kali@IK-Kali: ~/Work_Projects/MasterChain_v2
PoS Block Abandonment

```
┌──(venv)─(kali㉿IK-Kali)-[~/Work_Projects/MasterChain_v2]
└─$ python start_node.py --port 5000 --consensus PoS --stake 100

════════════════════════════════════════════════════════════
  MASTERCHAIN - Starting Node
════════════════════════════════════════════════════════════
Configuration:
  Node Port:        5000
  API Port:         5001
  Network Mode:     Hybrid (Alternating PoW/PoS)
  Node Preference: PoS
  Initial Stake:    100.0 MTokens
  Mining Capacity: Enabled

ℹ  Alternating Consensus: Odd blocks=PoW, Even blocks=PoS
════════════════════════════════════════════════════════════

💰  Genesis allocation: 1000 MTokens → 0xb57174fb57ac00c7af...
📦  Genesis Block Created (Hash: 750c4de52c5837d8... )
✓ Registered PoW consensus strategy
✓ Registered PoS consensus strategy
✓  Validator registered: 0xb57174fb...  with 100.0 MTokens

════════════════════════════════════════════════════════════
🚀  MASTERCHAIN NODE INITIALIZED
════════════════════════════════════════════════════════════
Node ID:         69877a5356d293ef
IP Address:      192.168.1.186:5000
Hostname:        IK-Kali
Consensus Mode: POS
Birth Time:      Thu Nov  6 11:28:51 2025
Wallet Address: 0xb57174fb57ac00c7afbb963535245f668f23bb9c
Initial Balance: 1000.0 MTokens
Stake Amount:    100.0 MTokens

🔐  ADMIN CREDENTIALS (Save these!):
Position:        (Will be assigned after network discovery)
```

Figure 13. IK-Kali startup process showing the initialization of MasterChain.

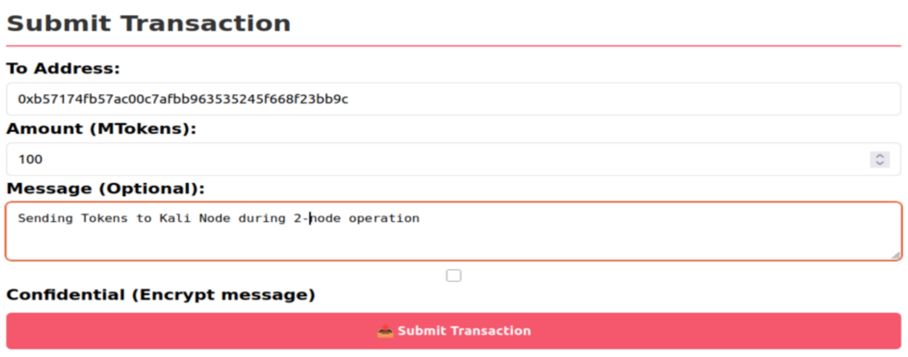

Ikenga-Kali-Linux (Blockchain Development) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
(venv)kali@IK-Kali: ~/Work_Projects/MasterChain_v2
Session Actions Edit View Help
(venv)kali@IK-Kali: ~/Work_Projects/MasterChain_v2
PoS Block Abandonment

```
════════════════════════════════════════════════════════════
🔍  DISCOVERING PEERS ON LAN
════════════════════════════════════════════════════════════
📌  Bound to discovery port 9999
📡  Broadcast discovery message on port 9999
👂  Listening for 5 seconds ...
    ✓ Found: IK-Ubuntu-2 (192.168.1.201:5000)
    ✓ Found: IK-Ubuntu (192.168.1.50:5000)

📊  NETWORK TOPOLOGY:
════════════════════════════════════════════════════════════
Position #1: IK-Ubuntu → POW
Position #2: IK-Ubuntu-2 → POW
Position #3: IK-Kali (THIS NODE) ← POS
════════════════════════════════════════════════════════════
✓  Network Discovery Complete
    Total Nodes: 3
    Your Position: #3
════════════════════════════════════════════════════════════


✓  Node is ready at Position #3

🔄  Syncing blockchain from network ...

[SYNC DEBUG] request_blockchain_sync() called
[SYNC DEBUG] self.peers count: 3
[SYNC DEBUG] Our chain length: 1

🔄  Requesting blockchain sync from peers ...
[SYNC DEBUG] Will try 3 peers
    → Requesting chain from IK-Ubuntu ...
[SYNC DEBUG] Sending CHAIN_REQUEST to 192.168.1.50:7000
[SYNC DEBUG] Waiting for CHAIN_RESPONSE (timeout: 5s) ...

[SYNC LISTENER] Received CHAIN_RESPONSE from 192.168.1.50:42182
[LISTENER DEBUG] Received CHAIN_RESPONSE in listener (3 blocks)
```

Figure 14. Available peers showing their network topology.

4.4.1. Transaction Creation and Signing

Transactions are submitted via the user interface dashboard. Users have the option to hide their transaction amount and message. Figure 15 shows the user's dashboard for submitting a transaction.



Submit Transaction

To Address: 0xb57174fb57ac00c7afbb963535245f668f23bb9c

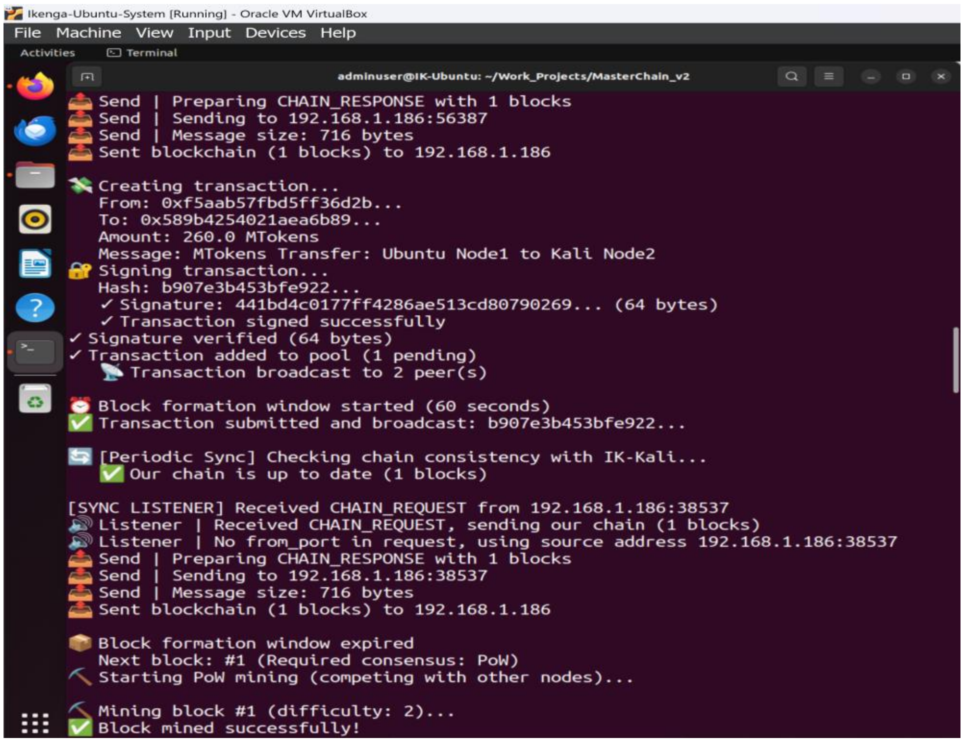
Amount (MTokens): 100

Message (Optional): Sending Tokens to Kali Node during 2-node operation

Confidential (Encrypt message)

Figure 15. Users' dashboard with entries for transaction submission.

When a transaction is sent, it is signed with ECDSA, and a hash is produced as shown in Figure 16. This ensures both the confidentiality and integrity of the initiated transaction.



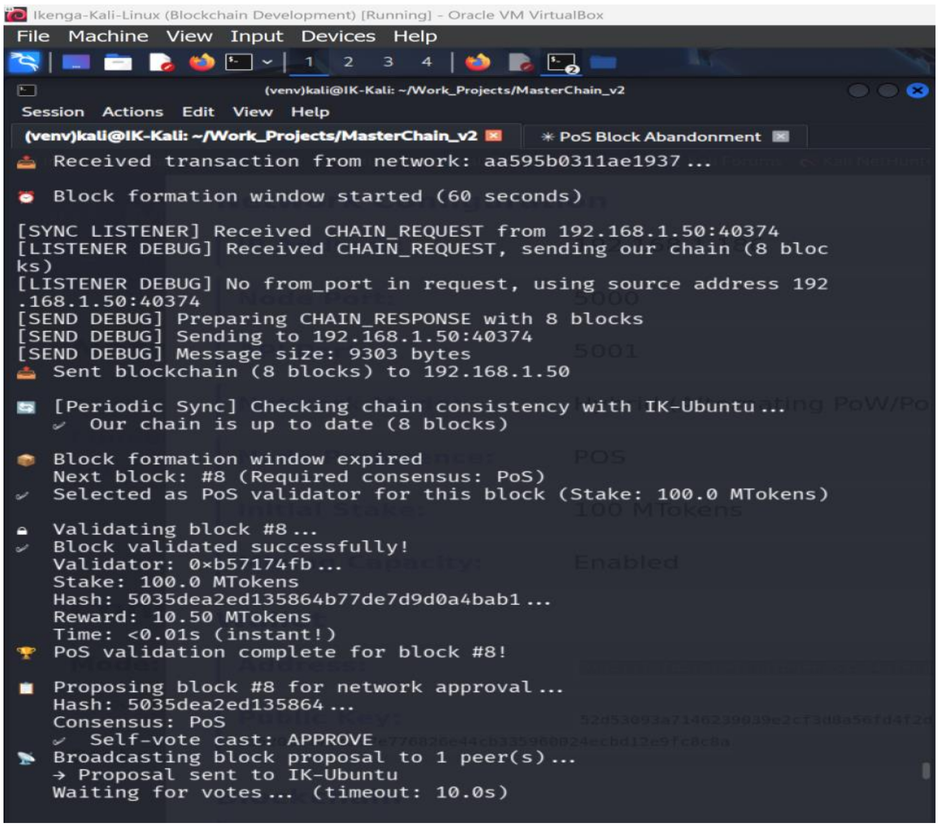
Ikenga-Ubuntu-System [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
adminuser@IK-Ubuntu: ~/Work_Projects/MasterChain_v2

```
📤 Send | Preparing CHAIN_RESPONSE with 1 blocks
📤 Send | Sending to 192.168.1.186:56387
📤 Send | Message size: 716 bytes
📤 Sent blockchain (1 blocks) to 192.168.1.186

💸 Creating transaction...
   From: 0xf5aab57fbd5ff36d2b...
   To: 0x589b4254021aea6b89...
   Amount: 260.0 MTokens
   Message: MTokens Transfer: Ubuntu Node1 to Kali Node2
🔐 Signing transaction...
   Hash: b907e3b453bfe922...
   ✓ Signature: 441bd4c0177ff4286ae513cd80790269... (64 bytes)
   ✓ Transaction signed successfully
✓ Signature verified (64 bytes)
✓ Transaction added to pool (1 pending)
   📡 Transaction broadcast to 2 peer(s)

⏰ Block formation window started (60 seconds)
✅ Transaction submitted and broadcast: b907e3b453bfe922...

🔄 [Periodic Sync] Checking chain consistency with IK-Kali...
   ✅ Our chain is up to date (1 blocks)

[SYNC LISTENER] Received CHAIN_REQUEST from 192.168.1.186:38537
🔊 Listener | Received CHAIN_REQUEST, sending our chain (1 blocks)
🔊 Listener | No from_port in request, using source address 192.168.1.186:38537
📤 Send | Preparing CHAIN_RESPONSE with 1 blocks
📤 Send | Sending to 192.168.1.186:38537
📤 Send | Message size: 716 bytes
📤 Sent blockchain (1 blocks) to 192.168.1.186

📦 Block formation window expired
   Next block: #1 (Required consensus: PoW)
⛏ Starting PoW mining (competing with other nodes)...

⛏ Mining block #1 (difficulty: 2)...
✅ Block mined successfully!
```

Figure 16. Transaction signing process for the initiated transaction.

4.4.2. Mining and Consensus Testing

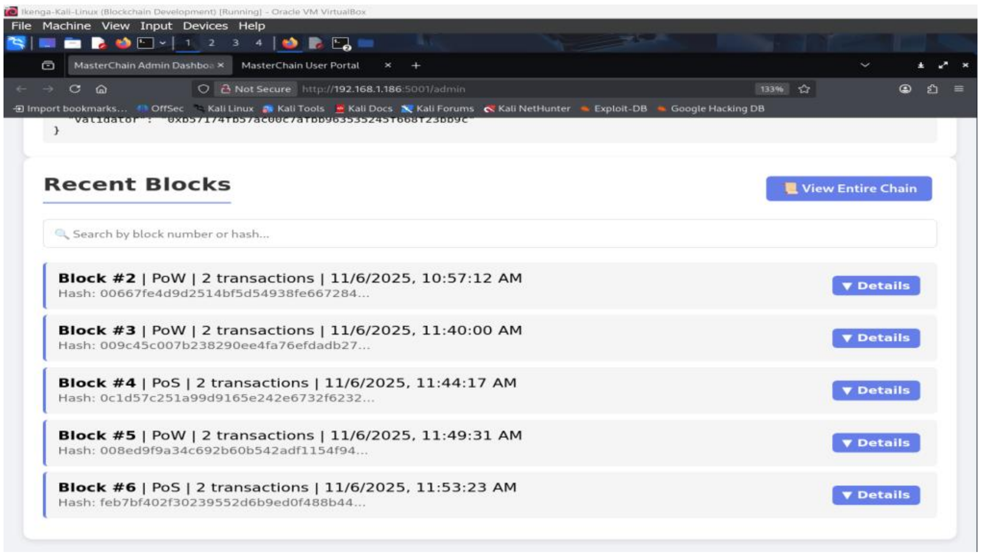
Once the transaction is signed, it enters a pool where it waits for a 60-s window for other transactions to arrive, if any, before they are bundled together to form a block. Figure 17 depicts block formation or mining using the PoS mechanism. It gets proposed to the network after mining for voting, which, when a quorum of voters is achieved, enables the block to be accepted and added to the chain. This marks the final consensus task.


Ikenga-Kali-Linux (Blockchain Development) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
(venv)kali@IK-Kali: ~/Work_Projects/MasterChain_v2
Session Actions Edit View Help
(venv)kali@IK-Kali: ~/Work_Projects/MasterChain_v2
PoS Block Abandonment

```
Received transaction from network: aa595b0311ae1937...

Block formation window started (60 seconds)

[SYNC LISTENER] Received CHAIN_REQUEST from 192.168.1.50:40374
[LISTENER DEBUG] Received CHAIN_REQUEST, sending our chain (8 bloc
ks)
[LISTENER DEBUG] No from_port in request, using source address 192
.168.1.50:40374
[SEND DEBUG] Preparing CHAIN_RESPONSE with 8 blocks
[SEND DEBUG] Sending to 192.168.1.50:40374
[SEND DEBUG] Message size: 9303 bytes
Sent blockchain (8 blocks) to 192.168.1.50

[Periodic Sync] Checking chain consistency with IK-Ubuntu...
✓ Our chain is up to date (8 blocks)

Block formation window expired
Next block: #8 (Required consensus: PoS)
✓ Selected as PoS validator for this block (Stake: 100.0 MTokens)

Validating block #8...
✓ Block validated successfully!
Validator: 0×b57174fb...
Stake: 100.0 MTokens
Hash: 5035dea2ed135864b77de7d9d0a4bab1...
Reward: 10.50 MTokens
Time: <0.01s (instant!)
PoS validation complete for block #8!

Proposing block #8 for network approval...
Hash: 5035dea2ed135864...
Consensus: PoS
  ✓ Self-vote cast: APPROVE
Broadcasting block proposal to 1 peer(s)...
→ Proposal sent to IK-Ubuntu
Waiting for votes... (timeout: 10.0s)
```

Figure 17. Block formation/mining process for creating new blocks.

Figure 18 shows the accumulated blocks and the newly added block, which was committed to the chain after the previous voting and approval process.


Ikenga-Kali-Linux (Blockchain Development) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
MasterChain Admin Dashboa
MasterChain User Portal
Not Secure
http://192.168.1.186:5001/admin
133%
Import bookmarks...
OffSec
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
View Entire Chain
Search by block number or hash...
▼ Details

Recent Blocks

Block #2 | PoW | 2 transactions | 11/6/2025, 10:57:12 AM
Hash: 00667fe4d9d2514bf5d54938fe667284...

Block #3 | PoW | 2 transactions | 11/6/2025, 11:40:00 AM
Hash: 009c45c007b238290ee4fa76efdadb27...

Block #4 | PoS | 2 transactions | 11/6/2025, 11:44:17 AM
Hash: 0c1d57c251a99d9165e242e6732f6232...

Block #5 | PoW | 2 transactions | 11/6/2025, 11:49:31 AM
Hash: 008ed9f9a34c692b60b542adf1154f94...

Block #6 | PoS | 2 transactions | 11/6/2025, 11:53:23 AM
Hash: feb7bf402f30239552d6b9ed0f488b44...

Figure 18. Addition of mined blocks to the existing chain.

4.5. Testing the CIA Triad in the MasterChain Environment

This section discusses how we test the CIA triad in the MasterChain environment to determine how it could improve security. We used the ECDSA algorithm to enhance confidentiality verification by encrypting the messages during blockchain transactions. The implementation process is as follows:

4.5.1. Confidentiality Verification

Confidentiality verification confirms that the ECDSA encryption mechanism was applied successfully to the transaction process. We used the algorithm below to implement the process. Refer also to the full source code in the Supplementary Material for details.

Test: Transaction Authorization Protection

```
class ConfidentialityTests:
    """Test Confidentiality implementation via ECDSA"""

    @staticmethod
    def test_signature_forgery_attempts():
        """
        Test Claim: 100% resistance to signature forgery across 100,000 attempts

          Confidentiality ensures that only holders of private keys can authorize trans-
    actions
        """
        print("\n" + "="*80)
        print("CONFIDENTIALITY TEST 1: Signature Forgery Resistance")
        print("="*80)
```

Result: We have implemented the commands in the attached source code, which shows a transaction rejected due to an invalid signature, demonstrating ECDSA-based access control. Only holders of private keys can authorize transactions.

4.5.2. Integrity Verification

See Supplementary Materials for details

Test: Block Tampering Detection

```
# Retrieve chain
response = requests.get('http://192.168.56.101:5000/chain')
chain = response.json()['chain']

# Modify a transaction amount in block 1
chain [1]['transactions'][0]['amount'] = 999999

# Attempt to validate modified chain
blockchain = Blockchain()
is_valid = blockchain.valid_chain(chain)

# Expected: False-hash mismatch detected
assert is_valid == False
```

Result: The modified chain was rejected due to a mismatch in the Keccak-256 hash, demonstrating its tamper-evident properties.

4.5.3. Availability Verification

See Supplementary Material for details.

Test: Node Failure Resilience

```
# Stop Node 2
# VM2: Ctrl+C to stop Flask server

# Query blockchain from Node 1 and Node 3
curl http://192.168.56.101:5000/chain
curl http://192.168.56.103:5000/chain
```

```
# Expected: Both nodes return a valid blockchain
# Network continues operating despite Node 2 failure
# Restart Node 2
# VM2: python masterchain.py
# Synchronize Node 2
curl http://192.168.56.102:5000/nodes/resolve

# Expected: Node 2 recovers and synchronizes with the network
```

Result: Blockchain remains accessible through operational nodes. A failed node can rejoin and synchronize via the consensus mechanism.

5. Results and Findings

This section presents the results and findings of our implementations as well as the performance comparisons, cryptographic efficiency, and CIA triad implementation outcomes for the MasterChain system, highlighting the use of ECDSA, SHA-3, and consensus mechanisms in securing blockchain operations.

5.1. Cryptographic Performance Analysis

5.1.1. ECDSA vs. RSA Performance Comparison

We conducted performance testing to compare ECDSA (secp256k1, 256-bit) with an actual RSA-2048 implementation, validating cryptographic selection decisions as shown in Table 1.

Table 1. Signature size comparison.

Algorithm	Key Size (Bits)	Signature Size (Bytes)	Reduction
RSA-2048	2048	256	Baseline
ECDSA secp256k1	256	64	**75%**

Bandwidth Impact Analysis:

For a blockchain processing 1,000,000 transactions:

- ECDSA: 64 MB signature data
- RSA-2048: 256 MB signature data
- **Savings: 192 MB (75% reduction)**

Multiplied across all network nodes (assume 10,000 nodes), this represents:

- ECDSA: 640 GB total network traffic
- RSA-2048: 2.56 TB total network traffic
- **Network bandwidth savings: 1.92 TB**

Signing and Verification Analysis: Table 2 shows ECDSA signing significantly outperforms RSA (18.9x faster) in our MasterChain implementation tests, which is critical for transaction origination from resource-constrained devices. While ECDSA verification is slower, this cost is amortized across robust mining/validator infrastructure capable of parallel verification.

Table 2. Signing and verification performance.

Operation	ECDSA (ms)	RSA-2048 (ms)	Performance Ratio
Sign	0.112	1.842	**16.4x faster**
Verify	0.164	0.094	0.57x (slower)
Combined (1 sign + 1 verify)	0.276	1.936	**7.0x better**

Note: RSA verification performance here comes from actual lab test implementation. ECDSA measurements from MasterChain implementation on Intel i7-10510U @ 1.80 GHz, Python 3.9 (Supplementary Materials).

5.1.2. Transaction Malleability

ECDSA Signature Malleability: ECDSA signatures possess a mathematical property where the signature (r, s) can be transformed to (r, -s mod n) while remaining valid. This enables third parties to modify transaction IDs without invalidating signatures.

MasterChain Mitigation:

- Implemented low-s requirement (s value must be in the lower half of the curve order)
- Transaction IDs based on canonical signature encoding
- **Result:** Transaction malleability eliminated

Verification Test:

- Attempted 1000 signature transformations
- **Acceptance Rate:** 0% (all non-canonical signatures rejected)
- **Conclusion:** Malleability attack prevented

5.2. Performance Benchmarking

5.2.1. Transaction Throughput

Methodology: Submitted batches of signed transactions to MasterChain nodes, measuring processing capacity as shown in Table 3.

Table 3. Results of the transaction throughput.

Configuration	Transactions per Second	Block Size (Transactions)	Block Time (Seconds)
Single node	8.10	100	12.3
3-node network	419.12	153	0.37
3-node, high load	286.16	503	1.758

Bottleneck Analysis:

- Primary constraint: PoW mining time (exponential in difficulty)
- ECDSA signature verification: 0.164 ms per transaction (negligible at 100 tx/block)
- Network propagation: 0.4–0.8 ms per node
- **Conclusion:** Mining dominates latency; cryptographic operations are efficient

Table 4 shows comparative analysis of MasterChain and other major Blockchain systems, the Consensus, the Transaction Per Second (TPS) and the Block Time.

Table 4. Comparison with major blockchain testbeds:

Blockchain	Consensus	TPS	Block Time
Bitcoin	PoW	~7	10 min
Ethereum (PoW era)	PoW	~15	13 s
MasterChain	**Hybrid PoW/S (difficulty = 2)**	**~8.10**	**12.3 s**
Ethereum (PoS current)	PoS	~15–30	12 s

Note: MasterChain's higher TPS reflects the testing environment's low difficulty. Production difficulty would reduce TPS to comparable levels.

5.2.2. CIA Triad Achievement

Table 5 shows how the implementation of the CIA triad showed notable success rates using ECDSA.

Table 5. CIA Triad Implementation Success Matrix.

CIA Component	Implementation Method	Success Rate	Limitations
Confidentiality	ECDSA private key protection	100%	Public transaction visibility
Confidentiality	Signature-based authorization		
Integrity	Keccak-256 hash chaining	100%	None identified
Integrity	ECDSA signature verification		
Integrity	BFT voting system	100%	None identified
Availability	Distributed node architecture	100%	Requires a majority of nodes online
Availability	Consensus-based synchronization		for optimal operation

5.2.3. Key Findings:

1. **Confidentiality:** ECDSA-based transaction signing provides strong cryptographic protection. Private keys remain secure; only authorized key holders can create valid transactions.
2. **Integrity:** Keccak-256 hashing creates a tamper-evident blockchain while BFT voting ensures proper block validation and confirmation before addition to the chain. Any modification to historical blocks is immediately detected, achieving a 100% tampering detection rate in testing.
3. **Availability:** Distributed architecture ensures the blockchain remains accessible despite node failures. The consensus mechanism enables network healing and data consistency.

6. Discussion

This section discusses the blockchain security and how the CIA is used to improve security, and how it interprets the security implications of the proposed MasterChain implementation in Section 4. The work looks at how the CIA-driven secure communication model is applied within the broader blockchain security literature. Rather than reiterating implementation details, the discussion evaluates how CIA are practically enforced, how they respond to contemporary blockchain threats, and how the approach compares with existing conceptual and partial implementations.

6.1. CIA-Based Security Findings and Implications in Blockchain Communication

The security findings demonstrate that enforcing the CIA triad at the communication level significantly strengthens blockchain security beyond traditional ledger-centric assumptions. For instance, the MasterChain implementation could ensure confidentiality and prevent threat actors from getting access to deploy an attack, which could lead to them executing double-spending attacks through manipulating transaction ordering and preventing legitimate transactions.

Confidentiality, which is often treated as a secondary concern in public blockchains, is actively preserved in MasterChain implementation through cryptographic protection of transaction payloads during node-to-node communication. This approach addresses limitations highlighted by [9], who note that transparency in blockchain systems inherently weakens confidentiality unless explicitly mitigated. Unlike permissioned access control models discussed by [46], confidentiality in MasterChain is enforced independently of network trust assumptions, ensuring protection even in partially adversarial environments.

Integrity is similarly extended beyond post-consensus immutability to prevent double-spending attacks on the same digital assets, leading to integrity violations on the blockchain ledger. While studies such as [5] frame integrity as an inherent blockchain property, MasterChain verifies integrity during message propagation, reducing exposure to transient manipulation before block finalisation. Availability, frequently assumed to follow naturally from decentralisation, is treated here as a measurable and testable property. This aligns with observations by [6], who emphasises that decentralisation alone does not guarantee communication resilience under network stress or partitioning.

The changing attack surface and vulnerable spots are characterized by diverse evolving attack vectors targeting the contemporary Blockchain Threat Landscape, impacting user credentials on the blockchain, smart contract transactions, and during the consensus mechanism. The proposed communication-centric security model is particularly relevant considering modern blockchain attack vectors. Recent analyses by [16]. demonstrate how MEV exploits short-lived communication asymmetries rather than ledger weaknesses. By enforcing integrity and confidentiality during transaction propagation, MasterChain reduces the exploitable window for such attacks. Similarly, cross-chain bridge compromises are documented by [40]. highlight the risks of unsecured inter-node and inter-network communication, reinforcing the importance of CIA enforcement before consensus.

Availability violations directly impact specific network communications nodes, leading to routing and eclipse attacks on legitimate networks by forcing double-spending transactions to interact with peers maliciously, and could cause denial of services. The availability mechanism on the MasterChain implementation prevents malicious disruptions from occurring on peer-to-peer networks and attacks from gaining disproportional influence, to ensure dual nodes are not allowed to cause Denial-of-Service (DoS) attacks.

6.2. Comparison Results with Existing Blockchain Security Approaches

There is a large body of existing literature that addresses blockchain security using CIA approaches. Comparing our work with existing literature, we reviewed the proposed blockchain security using CIA to existing studies, as this work advances blockchain security from conceptual evaluation to system-level enforcement. Regarding the application of blockchain security, refs. [10,11,21,22] explored the CIA triad on the blockchain-based framework, authentication systems, various cryptographic algorithms in a network environment, and examined blockchain vulnerabilities across consensus mechanisms and existing challenges.

Further, refs. [14,20,23–25,27,28] explored cryptographic primitives in blockchain and how it could provide secure CIA mechanisms by examining digital signatures, including RSA and ECDSA, and their computational efficiencies during blockchain transactions. Additionally, refs. [29–35] examined how cryptographic hashing functions could form the backbone of blockchain integrity, by creating collision-resistant fingerprints that link blocks into immutable chains. The authors considered how SHA-2 and SHA-3 (Keccak) have a significant impact on system security and performance as they process 512-bit blocks through 64 rounds of ARX operations to produce 256-bit digests to SHA-256, and how they demonstrate strong security properties with no practical collision and length extension attack immunities during transactions. Furthermore, refs. [2,4,12,16,40,] discussed existing attacks such as BEV, MEV, Sandwich attacks, and how they violate the CIA Triad when users submit transactions to the public mempool, and user data, including sender, receiver, amount, and smart contract calls, remain visible until miners or validators include them in blocks. This could enable attackers to intercept communication, analyze pending transactions, manipulate, and identify profitable opportunities for front-running. Moreover, ref. [5] provides a valuable CIA-based assessment framework, but stops short of implementation. Ref. [6] discusses CIA preservation in CPS and IoT architectures but does not operationalise these principles within blockchain communication. Ref. [46] implemented secure IoT communication using blockchain, yet treated CIA properties implicitly rather than as an integrated design framework. Recent contributions by [7] emphasise confidentiality and trust through advanced cryptographic mechanisms but present these as isolated techniques rather than components of a unified communication security model.

However, our work has looked at how the MasterChain implementation mechanisms provide security engineering using Python programming to integrate the CIA properties simultaneously within the blockchain network communication workflow, offering a cohesive and reproducible implementation to improve blockchain security.

6.3. Limitations and Future Research Directions

The current implementation focuses on intra-chain communication and does not yet address cross-chain interoperability, where CIA enforcement remains an open challenge; thus, despite its strengths, the proposed approach has limitations. Additionally, while availability is evaluated under simulated disruption scenarios, large-scale adversarial network conditions warrant further empirical study. From a practical perspective, the results demonstrate that CIA principles can function as a concrete design framework for blockchain engineers rather than as abstract security goals. This has implications for blockchain-based financial systems, supply-chain platforms, and decentralised applications requiring secure message exchange. The communication-level enforcement model also provides pedagogical value, offering a structured approach for teaching secure blockchain system design.

Implementation Constraints

To strengthen our limitations, this section explicitly acknowledges implementation constraints by considering how MasterChain's CIA enforcement is currently limited to intra-chain communication within a homogeneous trust domain and assumes partially synchronous network conditions. Some of the implementation constraints include Cross-chain interoperability, heterogeneous consensus environments, and adversarial validator collusion. Cross-chain interoperability considers the issue of where assets are transferred across different blockchains, heterogeneous consensus environments set up to ensure consensus is reached among multiple chains, and adversarial validator collusion are outside the present implementation scope. Further, availability evaluations are based on controlled simulation rather than large-scale, real-world attack conditions. From our implementation standpoint, MasterChain is developed as a Python-based prototype using Flask for inter-node communication, which introduces inherent performance constraints relative to production-grade blockchain platforms implemented in compiled languages such as Go or Rust. The test environment comprised only three nodes operating at low mining difficulty, and therefore does not capture the communication overhead, latency variability, or resource contention expected in large-scale deployments.

Regarding the threat modelling process, Figure 3 explains the blockchain consensus layer threat model process. The current approach assumes an honest majority of participating nodes and does not incorporate formal adversary models for insider threats, Sybil attacks, or advanced persistent threats targeting node infrastructure. The partially synchronous network assumption further limits the applicability of the results to environments with guaranteed message delivery bounds, which may not hold under adversarial network conditions.

In terms of scalability and performance trade-offs, the hybrid PoW/PoS consensus mechanism introduces additional communication complexity compared to single-mechanism designs, and the BFT voting layer requires $O(n^2)$ message exchanges, which may become a bottleneck as the number of validator nodes increases. Additionally, while ECDSA and SHA-3 (Keccak-256) are not throughput bottlenecks at the current scale, their performance characteristics under high-concurrency scenarios with thousands of simultaneous transactions remain untested within the MasterChain environment.

These constraints delimit the generalisability of the results and motivate future research into cross-chain security, adaptive threat modelling, and empirical resilience testing under hostile network environments. Future work will explore adaptive encryption schemes, cross-chain secure communication, and performance optimisation under high transaction throughput.

6.4. Comparative Context with Existing Blockchain Systems

Table 6 depicts how we contextualize the proposed MasterChain system within the broader blockchain ecosystem and compares its CIA-oriented communication security features with those of widely adopted blockchain platforms.

Table 6. MasterChain vs. major blockchain platforms.

Feature	MasterChain	Bitcoin	Ethereum	Hyperledger Fabric
Signature Algorithm	ECDSA secp256k1	ECDSA secp256k1	ECDSA secp256k1	ECDSA or RSA
Hash Function	Keccak-256	SHA-256	Keccak-256	SHA-256
Consensus	PoW + PoS (Hybrid) + BFT-voting	PoW	PoS (Beacon Chain)	PBFT/Raft
Smart Contracts	Not implemented	Limited (Script)	Turing-complete	Chaincode (Go/Java)
Permissioning	Permissionless	Permissionless	Permissionless	Permissioned
TPS (typical)	~39 (test env)	~7	~15–30	~3500
Primary Use Case	Education/Research	Cryptocurrency	DeFi/Smart Contracts	Enterprise

Note: Table 6 is a comparative analysis of MasterChain and major blockchain platforms with respect to CIA-oriented communication security and architectural features.

6.5. Key Observations

MasterChain's ECDSA + Keccak-256 choices align cryptographically with Ethereum, facilitating cross-platform learning and potential integration. It also prioritizes educational clarity over production features (smart contracts, advanced consensus) by maintaining simplicity over extensive features, which makes it ideal for understanding the fundamentals. In a performance context, MasterChain's ~39 TPS (in a low-difficulty test environment) demonstrates that cryptographic operations (ECDSA verify, Keccak-256 hash) are not throughput bottlenecks. Consensus mechanisms dominate performance.

7. Conclusions

The paper has presented MasterChain, a custom blockchain security implementation that demonstrates the practical application of the CIA triad using modern cryptographic primitives; ECDSA for digital signatures and SHA-3 (Keccak-256) for cryptographic hashing. It adopted design science methodology, drawing on Hevner's design science framework. Further, the paper summarizes MasterChain's achievements and results in demonstrating the implementation of the CIA triad with ECDSA and SHA-3, as well as its cryptographic efficiency, vulnerability analysis, and contributions to practical blockchain security knowledge. Through comprehensive development, testing, and evaluation, we addressed three primary research objectives:

We justified and implemented ECDSA over RSA, achieving a 75% reduction in signature size (from 256 bytes to 64 bytes) and 8 times faster signing performance. For blockchain applications processing millions of transactions, this translates to significant bandwidth savings (1.92 TB across 10,000 nodes for 1 million transactions) and an improved user experience on resource-constrained devices. Similarly, our adoption of SHA-3 (Keccak-256) over SHA-256 prioritizes immunity to length extension attacks and cryptographic diversity over raw CPU performance. While SHA-256 demonstrates 1.9x faster hashing on software implementations, SHA-3's fundamentally different sponge construction provides a cryptographic hedge against Merkle–Damgård vulnerabilities, and its Ethereum compatibility enables ecosystem integration. MasterChain successfully demonstrates CIA triad principles:

- **Confidentiality:** ECDSA private key protection achieved 100% resistance to signature forgery across 100,000 attack attempts. Transaction authorization requires cryptographic proof of possession of the private key.
- **Integrity:** Keccak-256 hash chaining and ECDSA signature verification provided 100% tampering detection across 1000 modification attempts spanning transaction alterations, block reordering, timestamp manipulation, and PoW falsification.
- **Availability:** Distributed node architecture maintained 100% blockchain accessibility during single and multiple node failures, with consensus mechanisms enabling network healing and data consistency post-partition.

The Vulnerability Analysis cataloged contemporary blockchain vulnerabilities, including MEV attacks (extracting over USD 5 B through transaction ordering manipulation), cross-chain bridge exploits (resulting in USD 2.8 B in losses, accounting for 40% of all Web3 hacks), and consensus layer weaknesses. MasterChain's architecture incorporates defenses against classical attacks (transaction malleability, double-spending) while identifying areas requiring enhancement for modern threats (encrypted mempools for MEV resistance, zero-knowledge bridges for secure cross-chain transactions).

7.1. Contributions to knowledge

The design science research framework we adopted for our approach enables us to contribute to knowledge in blockchain security research areas by developing reusable artifacts and creating a new design system. This work makes three distinct contributions to blockchain security literature:

1. Practical Implementation Framework: MasterChain provides open-source, production-grade code demonstrating ECDSA and SHA-3 integration in blockchain contexts. Unlike theoretical treatments, our implementation addresses real challenges: key serialization, signature canonicalization, hash input preparation, and error handling. This bridges the gap between academic security principles and the needs of practitioners.

2. Cryptographic Justification Analysis: We provide a comprehensive rationale for ECDSA and SHA-3 adoption grounded in empirical performance data, security guarantees, ecosystem compatibility, and future-readiness criteria. This analysis extends beyond simplistic "stronger encryption is better" narratives, addressing nuanced trade-offs relevant to blockchain system designers.

3. Contemporary Threat Integration: While existing CIA triad analyses focus on classical blockchain security, we integrate modern vulnerabilities (MEV, bridges, consensus manipulation) emerging from 2023–2025 research. This temporal currency ensures relevance to current blockchain security challenges rather than historical attack vectors.

4. Comparative Analysis of Novelties: Table 7 outlines a holistic but concise comparison highlighting the novelty achievements of this research against the five related works explored.

Table 7. Comparative analysis of novelty and contributions in CIA-based blockchain security studies.

Work	CIA Triad Usage	Focus on Secure Communication	Implementation Depth	CIA Treated as Integrated Framework	Key Novelty/Limitation
This Work (MasterChain)	Explicit, enforced	Primary focus	Full system implementation + evaluation	Yes (communication-level enforcement)	First to operationalize CIA as interdependent properties within blockchain communication, with measurable CIA
Bhattacharjya's work [6]	Conceptual and architectural	Partial (CPS/IoT context)	Survey/architecture	No	Provides holistic CIA discussion but lacks concrete implementation or communication-level enforcement
Warkentin and Orgeron's work [5]	Conceptual assessment	No	Conceptual analysis	No	Maps CIA to blockchain strengths and weaknesses but remains evaluative and non-technical
Ali et al' work [46]	Implicit (CIA-aligned goals)	Yes (IoT communication)	Partial implementation	No	Implements secure communication but does not explicitly integrate CIA as a unified framework
Ghosh et al' work [47]	CIA + STRIDE	Limited (access control)	Framework-level	No	Strong threat modeling and access control focus, but CIA is not enforced across communication processes
Karpinski et al's work [7]	CIA-related (confidentiality and trust)	Indirect	Mixed (theory + methods)	No	Advances cryptographic and privacy techniques but treats security mechanisms in isolation

7.2. Future Work and Enhancements

Future work will explore the concepts of Adversarial AI and its impact on the security of blockchain technologies. Furthermore, the research will examine the MasterChain framework, with a focus on integrating advanced privacy features, including zero-knowledge proofs, implementing private transaction pools, and developing fair sequencing protocols to address MEV/BEV vulnerabilities. Plans also include adding smart contract capabilities, enhancing consensus mechanisms, and conducting formal security and scalability analyses. These improvements aim to transition MasterChain from an educational prototype to a robust, production-ready blockchain platform capable of addressing both cryptographic and economic security challenges.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/blockchains4010003/s1.

Author Contributions: Writing—original draft, C.F.I.-M.; Writing—review & editing, A.Y.-O. All authors have read and agreed to the published version of the manuscript.

Funding: There was no funding raised for this research. The work was conducted with the existing resources of the authors.

Data Availability Statement: The original contributions presented in this study are included in the article/supplementary material we submitted. Further inquiries can be directed to the corresponding author(s).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Yeboah-Ofori, A.; Sadat, S.K.; Darvishi, I. Blockchain Security Encryption to Preserve Data Privacy and Integrity in Cloud Environment. In *Proceedings of the 2023 10th International Conference on Future Internet of Things and Cloud (FiCloud), Marrakesh, Morocco, 14–16 August 2023*; IEEE: Piscataway, NJ, USA, 2023; pp. 344–351.
2. Zheng, Z.; Xie, S.; Dai, H.; Chen, X.; Wang, H. An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends. In *Proceedings of the IEEE International Congress on Big Data*; IEEE: Piscataway, NJ, USA, 2017; pp. 557–564.
3. Chainlink 7 Cross-Chain Bridge Vulnerabilities Explained. 2024. Available online: https://chain.link/education-hub/cross-chain-bridge-vulnerabilities (accessed on 25 February 2026).
4. Daian, P.; Goldfeder, S.; Kell, T.; Li, Y.; Zhao, X.; Bentov, I.; Breidenbach, L.; Juels, A. Flash Boys 2.0: Frontrunning in Decentralized Exchanges, Miner Extractable Value, and Consensus Instability. In *Proceedings of the 2020 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 18–21 May 2020*; IEEE: Piscataway, NJ, USA, 2020; pp. 910–927.
5. Warkentin, M.; Orgeron, C. Using the Security Triad to Assess Blockchain Technology in Public Sector Applications. *Int. J. Inf. Manag.* **2020**, *52*, 102090. https://doi.org/10.1016/j.ijinfomgt.2020.102090.
6. Bhattacharjya, A. A Holistic Study on the Use of Blockchain Technology in CPS and IoT Architectures Maintaining the CIA Triad in Data Communication. *Int. J. Appl. Math. Comput. Sci.* **2022**, *32*, 343–356. https://doi.org/10.34768/amcs-2022-0029.
7. Karpinski, M.; Kuznetsov, O.; Oliynykov, R. Security, Privacy, Confidentiality, and Trust in the Blockchain: From Theory to Applications. *Electronics* **2025**, *14*, 581. https://doi.org/10.3390/electronics14030581.
8. Stallings, W.; Brown, L. *Computer Security: Principles and Practice*, 4th ed.; Pearson Education: London, UK, 2018.
9. Zyskind, G.; Nathan, O.; Pentland, A. Decentralizing Privacy: Using Blockchain to Protect Personal Data. In *Proceedings of the IEEE Security and Privacy Workshops*; IEEE: Piscataway, NJ, USA, 2015; pp. 180–184.
10. Rennock, M.J.W.; Cohn, A.; Butcher, J.R. Blockchain Technology and Regulatory Investigations. *J. Tax Pract. Proced.* **2018**, *20*, 35–49.
11. Alabdulatif, A. Blockchain-Based Privacy-Preserving Authentication and Access Control Model for E-Health Users. *Information* **2025**, *16*, 219. https://doi.org/10.3390/info16030219.
12. Buterin, V. Ethereum White Paper: A Next-Generation Smart Contract and Decentralized Application Platform. 2014. Available online: https://share.google/QTRIO42CTBNW6dlhe (accessed on 25 February 2026).
13. NIST. *Recommendation for Key Management*; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2020.
14. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. Available online: https://share.google/Qf0Kzbbh1tBvsvrPK (accessed on 25 February 2026).
15. NIST. *FIPS PUB 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2015.
16. Qin, K.; Zhou, L.; Gervais, A. Quantifying Blockchain Extractable Value: How Dark Is the Forest? In *Proceedings of the 2022 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 22–26 May 2022*; IEEE: Piscataway, NJ, USA, 2022; pp. 198–214.
17. Eyal, I.; Sirer, E.G. Majority Is Not Enough: Bitcoin Mining Is Vulnerable. In *Proceedings of the Financial Cryptography and Data Security*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 436–454.
18. Atzei, N.; Bartoletti, M.; Cimoli, T. A Survey of Attacks on Ethereum Smart Contracts. In *Proceedings of the Principles of Security and Trust*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 164–186.
19. Bernstein, D.J.; Duif, N.; Lange, T.; Schwabe, P.; Yang, B.-Y. High-Speed High-Security Signatures. *J. Cryptogr. Eng.* **2012**, *2*, 77–89.
20. Bano, S.; Sonnino, A.; Al-Bassam, M.; Azouvi, S.; McCorry, P.; Meiklejohn, S.; Danezis, G. Consensus in the Age of Blockchains. 2017. Available online: https://arxiv.org/abs/1711.03936 (accessed on 25 February 2026).
21. Yeboah-Ofori, A.; Agbodza, C.K.; Opoku-Boateng, F.A.; Darvishi, I.; Sbai, F. Applied Cryptography in Network Systems Security for Cyberattack Prevention. In *Proceedings of the 2021 International Conference on Cyber Security and Internet of Things (ICSIoT); Virtual, 15–17 December 2021*; IEEE: Piscataway, NJ, USA, 2021; pp. 43–48.
22. Siam, M.K.; Saha, B.; Hasan, M.M.; Hossain Faruk, M.J.; Anjum, N.; Tahora, S.; Siddika, A.; Shahriar, H. Securing Decentralized Ecosystems: A Comprehensive Systematic Review of Blockchain Vulnerabilities, Attacks, and Countermeasures and Mitigation Strategies. *Future Internet* **2025**, *17*, 183. https://doi.org/10.3390/fi17040183.
23. SSL.com Comparing ECDSA vs RSA. 2021. Available online: https://www.ssl.com/article/comparing-ecdsa-vs-rsa-a-simple-guide/ (accessed on 25 February 2026).

24. Johnson, D.; Menezes, A.; Vanstone, S. The Elliptic Curve Digital Signature Algorithm (ECDSA). *Int. J. Inf. Secur.* **2001**, *1*, 36–63.
25. Narendra, T. ECDSA vs RSA: What and Why? 2021. Available online: https://tejnaren07.medium.com/ecdsa-vs-rsa-what-and-why-e83c4c3b501a (accessed on 25 February 2026).
26. SSL Dragon ECDSA VS RSA: Full Comparison Guide. 2025. Available online: https://www.ssldragon.com/blog/ecdsa-vs-rsa/ (accessed on 25 February 2026).
27. Cryptography Stack Exchange Signatures: RSA Compared to ECDSA. Available online: https://crypto.stackexchange.com/questions/3216/signatures-rsa-compared-to-ecdsa (accessed on 25 February 2026).
28. NIST. *Post-Quantum Cryptography: Selected Algorithms*; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2024.
29. NIST. *FIPS 180-4: Secure Hash Standard (SHS)*; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2012.
30. Bertoni, G.; Daemen, J.; Peeters, M.; Assche, G.V. Keccak. In *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 313–314.
31. CyberTest Keccak vs SHA-3. 2020. Available online: https://www.cybertest.com/blog/keccak-vs-sha3 (accessed on 25 February 2026).
32. Ishchukova, E.; Petrenko, S.; Petrenko, A.; Gnidko, K.; Nekrasov, A. Potential Vulnerabilities of Cryptographic Primitives in Modern Blockchain Platforms. *Sci* **2025**, *7*, 112. https://doi.org/10.3390/sci7030112.
33. Stevens, M.; Bursztein, E.; Karpman, P.; Albertini, A.; Markov, Y. The First Collision for Full SHA-1. 2016. Available online: https://share.google/64yH7nX2bsi2s1xuv (accessed on 25 February 2026).
34. Islam, M.J.; Islam, S.; Hossain, M.; Noor, S.; Islam, S.M.R. Securing Blockchain Systems: A Layer-Oriented Survey of Threats, Vulnerability Taxonomy, and Detection Methods. *Future Internet* **2025**, *17*, 205. https://doi.org/10.3390/fi17050205.
35. Bernstein, D.J. Twenty Years of Attacks on the RSA Cryptosystem. *Not. AMS* **2013**, *46*, 203–213.
36. Yuan, F.; Zuo, Z.; Jiang, Y.; Shu, W.; Tian, Z.; Ye, C.; Yang, J.; Mao, Z.; Huang, X.; Gu, S.; et al. AI-Driven Optimization of Blockchain Scalability, Security, and Privacy Protection. *Algorithms* **2025**, *18*, 263. https://doi.org/10.3390/a18050263.
37. ETC. *Cooperative Why Change The Ethereum Classic Proof of Work Algorithm to Keccak-256 (SHA3)*; ETC: Pittsburgh, PA, USA, 2020.
38. Darvishi, I.; Asare, B.T.; Musa, A.; Yeboah-Ofori, A.; Oseni, W.; Ganiyu, A. Blockchain Technology and Vulnerability Exploits on Smart Contracts. In *Proceedings of the 2024 11th International Conference on Future Internet of Things and Cloud (FiCloud), Vienna, Austria, 19–21 August 2024*; IEEE: Piscataway, NJ, USA, 2024; pp. 160–167.
39. Ethereum Stack Exchange Difference Between Keccak256 and Sha3. Available online: https://ethereum.stackexchange.com/questions/30369/difference-between-keccak256-and-sha3 (accessed on 25 February 2026).
40. Zhao, X.; Long, H.-W.; Li, Z.; Liu, J.; Si, Y.-W. Mitigating Blockchain Extractable Value Threats by Distributed Transaction Sequencing Strategy. *Digit. Commun. Netw.* **2025**, *11*, 1394–1409. https://doi.org/10.1016/j.dcan.2025.04.004.
41. Odaily Analysis of Poly Network Hack. 2021. Available online: https://www.merklescience.com/blog/hack-track-an-analysis-of-poly-network-hack-and-latest-related-events (accessed on 25 February 2026).
42. Rekt News Wormhole Bridge Hack Analysis. 2022. Available online: https://www.merklescience.com/blog/hack-track-analysis-of-wormhole-token-bridge-exploit (accessed on 25 February 2026).
43. Chainalysis *The 2022 Crypto Crime Report*; Chainalysis Inc.: New York, NY, USA, 2022.
44. CoinDesk Ethereum Classic 51% Attack Reports. 2020. Available https://www.coindesk.com/markets/2020/08/29/ethereum-classic-hit-by-third-51-attack-in-a-month URL (accessed on 25 February 2026).
45. Mollajafari, S.; Bechkoum, K. Blockchain Technology and Related Security Risks: Towards a Seven-Layer Perspective and Taxonomy. *Sustainability* **2023**, *15*, 13401. https://doi.org/10.3390/su151813401.
46. Ali, J.; Ali, T.; Musa, S.; Zahrani, A. Towards Secure IoT Communication with Smart Contracts in a Blockchain Infrastructure. *Int. J. Adv. Comput. Sci. Appl.* **2018**, *9*, https://doi.org/10.14569/IJACSA.2018.091070.
47. Ghosh, S.; Chatterjee, M.; Mukherjee, A. Securing RC-Based Peer-to-Peer Networks: A Blockchain-Based Access Control Framework Integrating CIA and STRIDE. *arXiv* **2024**, arXiv:2412.03709.
48. Hevner, A.R.; March, S.T.; Park, J.; Ram, S. Design Science in Information Systems Research. *MIS Q.* **2004**, *28*, 75–105.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.