A Novel Merging Framework for Homogeneous and Heterogeneous Blockchain Systems*

Liehuang Zhu[1], Sadaf Bukhari[1], Kashif Sharif[1], Fan Li[1], Shumaila Fardous[1], and Sujit Biswas[2]

[1] Beijing Institute of Technology, Beijing, China
{liehuangz, sadaf, kashif, fli, shumailafardous}@bit.edu.cn
[2] CITY University of London, London, United Kingdom
sujit.biswas@city.ac.uk

Abstract. This paper presents two solutions for merging diverse blockchain systems, Homogeneous Chain Consolidation (HCC) and Heterogeneous Multi-chain Integration Architecture (HMIA). As blockchain networks expand, integrating these disparate ledgers becomes crucial. HCC merges similar chains using common genesis blocks, enhancing efficiency by pooling validator resources. HMIA integrates different chains through meta-blocks, balancing compatibility and autonomy. We evaluate these approaches using consensus times, block sizes, and ledger growth. HCC improves throughput and efficiency, while HMIA prioritizes decentralized attributes. Use case-based evaluation shows that the proposed work can be seamlessly integrated with existing solutions, while the performance of implementation proves the low cost of resources in merged networks. Our frameworks offer insights into merging blockchains, addressing critical challenges in interconnected decentralized networks.

Keywords: Blockchain Networks · BC Merging · Distributed Ledger

1 Introduction

In dynamic blockchain technology applications, merging or unifying two independent blockchains into a single cohesive ecosystem becomes inevitable. This process of merging joins two formerly distinct and autonomous blockchain networks into a unified one [17]. Its main motivations include consolidating multiple blockchain communities, projects, developers, and resources into a unified ecosystem and scaling by combining computing power, network nodes, and transaction throughput. The merging process typically involves carefully coordinating the consensus mechanisms, network rules, and economic models to ensure a smooth transition and avoid issues like duplicate transactions, double-spending, or disagreements over the canonical chain history [10, 21].

* This work is supported by Beijing Natural Science Foundation (IS23056) and National Natural Science Foundation of China No. 62232002. Corresponding author: K. Sharif.

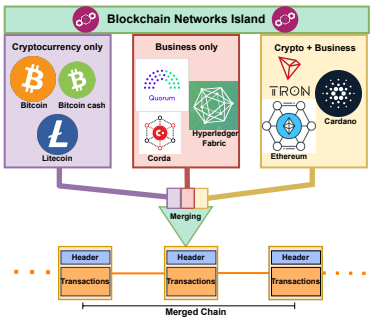


Fig. 1: Blockchain Merging Conceptualization.

Blockchain deployment has seen a significant increase in recent years. However, each blockchain network operates in isolation, possessing unique characteristics and goals that limit their potential benefits and utility, as illustrated in Figure 1. It visualizes the blockchain classification and merging concept, where independent and isolated chains can be merged into one. To maximize productivity and streamline communication, there is a growing imperative to foster interoperability among these networks. Blockchain interoperability is crucial for enabling the seamless exchange of information between different blockchains. However, achieving interoperability poses several challenges, including increased overhead in managing workflows, difficulties in coordinating transactions across different blockchains, heightened complexity, governance, security vulnerabilities arising from differences in consensus algorithms, challenges in developing bug-free smart contracts leading to security risks, heightened risk of double spending attacks, and the need to maintain data privacy [2, 15, 16]. Merging chains can have significant benefits, such as retaining decentralization, improving interoperability, consensus, and performance, and driving standardization across diverse applications. Merging offers a straightforward architecture instead of adding extra layers that could fail. The reasons for merging can vary depending on the specific use cases but often include consolidating resources, improving scalability, and fostering cross-chain interoperability to enhance the ecosystem's capabilities. For instance, Ethereum merged to reduce power consumption by 99.84-99.9996 % drastically [5]. Similarly, the upcoming merger of Klaytn and Finschia Mainnet in the second quarter of 2024 aims to boost scalability and token liquidity in critical Asian markets [12].

In this paper, our primary focus is on merging blockchain networks. It is important to note that interoperability differs from merging and is discussed later in the paper. We propose two distinct approaches for achieving the strategic merger of blockchain networks: Homogeneous Chain Consolidation (HCC) and Heterogeneous Multi-chain Integration Architecture (HMIA). The main contributions of this paper are as follows:

- We present the HCC algorithm that allows the joining of a secondary chain S_i to the primary chain P_0, such that the chains utilize similar consensus rules, transaction structures, and signatures henceforth.
- We propose a subsequent genesis block G_i where $i \geq 1$ is committed with a specialized consensus mechanism to bind the rules for the unification. This mechanism allows n secondary chains to be merged into C.
- We propose an HMIA algorithm that merges distinct blockchains with no common architectures. Here, two chains S_i and S_j where $i \neq j$ join into a single P_0 with a composite structure.
- We present a "merged block" M structure that bridges the gaps between diverse blockchain frameworks.
- We implement both proposed algorithms and show thorough analysis that the solutions are feasible and have negligible performance impact due to the consolidation process.

In the following sections, we first review existing research on blockchain merging. We then elaborate on the merging and interoperability needs and the design choices for different types of merges. Then, we present the proposed HCC and HMIA algorithms, followed by an evaluation of different use cases. In the end, we present the conclusion.

2 Related Works

Blockchain merging has attracted limited practical attention as the process is complicated. However, this does not mean the need for merging is not present. Authors in [4] introduce a blockgraph architecture based on Directed Acyclic Graph (DAG) to handle node mobility and network partitions in ad hoc networks, merging splits via reference blocks combining all previous chain history. Gai et al. also investigated block merging in a blockchain system based on a single chain in their study [8]. In order to reconstitute a single chain, they presented a new consensus process focused on DAGs used to reach an agreement on new blocks. This technique arranges and combines original blocks from a DAG structure. Moreover, existing literature proposes various solutions to modify chain data and challenge the immutable nature of the blockchain. For instance, the work on [6] introduces a redactable blockchain for Bitcoin, utilizing a voting-based consensus mechanism for chain alterations. Another study [14] presents a solution for altering transaction fields in the blockchain through the use of Chameleon hash. However, these solutions cannot be used for merging as this will destroy the time stamps, and the computational cost is unacceptable.

A forthcoming Mainnet Merge described in [12], between Finschia and Klaytn aims to establish Asia's premier blockchain ecosystem by integrating Ethereum and Cosmos technologies for exceptional compatibility and performance. This geo-spanning network plans to join key regions into a PDT token economy encompassing over 420 Dapps across Korea, Japan, Singapore, Taiwan, Vietnam, Thailand, and Abu Dhabi. In a study by Kapengut et al. [11], the Ethereum

network underwent The Merge switching to Proof of Stake (PoS) from Proof of Work (PoW) through a Beacon Chain after seven years. Miners did not transition into validators following the merge, leading to a 97% reduction in total block reward income. However, transaction fees for Ether increased by nearly 10%.

Existing literature primarily concentrates on merging blocks for various networks and consensus requirements. In contrast, our research targets explicitly merging chains within blockchain networks. Additionally, the immutable nature of the blockchain is being challenged, leading to the development of various data modification techniques such as redaction, pruning, updating, and erasing. These methods only partially modify contents, not fully merge separate chains. Instead, we design low-cost solutions for ledger combinations. We show straightforward ways of unifying blockchain networks through genesis blocks and meta-containers. Our proposed blockchain merging solutions show that it is possible to combine separate blockchain networks into consolidated systems, essentially overcoming the perception that blockchains cannot be altered after the fact.

3 Blockchain Merging

3.1 Interoperability vs Merging

We first clarify the difference in terminologies for interoperability and merging in blockchain networks. Interoperability in blockchain allows building application bridges for transferring assets and data across chains while retaining the independence of underlying networks through transmission protocols and semantic alignment [2, 7]. Contrarily, merging refers to fully integrating previously separate chains into one unified chain structure at the foundational protocol level in terms of consolidated consensus rules, cryptography, and validation semantics. Interoperability can be achieved, however, this requires application-level specialized interfaces [3]. Structural and semantic interoperability must be addressed using these interfaces. The higher-level applications creates a shell around them for unified working. This may be a workable system but requires more resources as the BC networks are still independent. Interoperability challenges encompass securely transferring assets across different decentralized networks, the need for custom bridging infrastructure, asset transfer, latency arising from locking and unlocking mechanisms, and the complexities of integrity checks to validate states due to differing consensus rules.

3.2 Classification and Critical Factors

Not all blockchains can be merged ideally. Blockchains can be categorized as cryptocurrency, business, or hybrid functions, with additional dimensions like permissions, smart contracts, and use cases. Cryptocurrency-focused blockchains (e.g., Bitcoin, Bitcoin Cash, Litecoin) focus on digital payments, whereas Ripple is a public permissioned example [10]. Business-centric blockchains (e.g.,

Table 1: Factors and properties of different blockchain networks.

No.	Blockchain	Category	Block Size	Hash & Digital sign.	SC Capability	Consensus Algo.
1	Bitcoin	Crypto currency only	80 B	SHA-256 - secp256k1 ECDSA	Very limited Bitcoin Script	PoW
2	Litecoin	Crypto currency only	80 B	SHA-256 - secp256k1 ECDSA	Very limited Litecoin Script	PoW
3	Bitcoin Cash	Crypto currency only	80 B	SHA-256 - secp256k1 ECDSA	Very limited Bitcoin Cash Script	PoW
4	Hyperledger Fabric	Business only	≈10.06 KB	ECDSA	Yes Chaincode	Kafka, Raft, PBFT, PoS
5	Corda	Business only	No block	EDD25199 with 512 hash	Yes Corda contracts	Raft - BFT
6	Quorum	Business only	512 B	Keccak-256 - secp256k1 ECDSA	Yes No specific name	Raft - Istanbul BFT
7	Ethereum	Hybrid	512 B	Keccak-256 - secp256k1 ECDSA	Yes No specific name	PoW moving towards PoS
8	Cardano	Hybrid	92 B	BLAKE2b - Ed25519	Yes Plutus Script	Ouroboros PoS
9	Tron	Hybrid	112 B	SHA-256 - secp256k1 ECDSA	Yes TRON bytecode	Delegated PoS

Hyperledger Fabric, Corda, Quorum) often provide supply chain and identity solutions using permissioned models. Hybrid ledgers (e.g., Ethereum, Cardano, Tron) support cryptocurrency and smart contracts for end-users and developers, with public and private channels [9, 13, 17]. Before merging, design decisions have to be made based on analyzing dimensions like consensus, data models, and governance aids in assessing blockchain compatibility. Blockchain networks often operate as isolated islands, each with its own technical and governance parameters. While some chains share common roots and can merge more seamlessly, connecting independent blockchains poses challenges. Before merging disparate networks, several critical factors must be considered, which are discussed and listed below in further detail in Table 1.

- Consensus Rules: Mechanisms by which transactions are validated, and blocks committed differ across chains,e.g., Bitcoin uses PoW and Ethereum is transitioning to PoS. Reconciling conflicting consensus rules is essential for BCs.
- Data Formats: Details like block size, state storage, supported transaction types, and contracts vary between platforms like Bitcoin, Ethereum, and Cosmos. These low-level technical formats must be unified.
- Cryptographic Primitives: Chains utilize different cryptographic hash functions (SHA-256, SHA-3) and digital signature schemes (ECDSA, EdDSA). Compatibility helps guard against hash collisions or failed verification.

4 HCC Framework

The Hybrid Chain Consolidation (HCC) algorithm smoothly merges compatible homogeneous blockchain networks, preserving complete transaction histories. Genesis block connects both chains' histories in the new unified chain, ensuring compatibility in technical formats, consensus algorithms, cryptographic primitives, virtual machine semantics, and permissions boundaries.

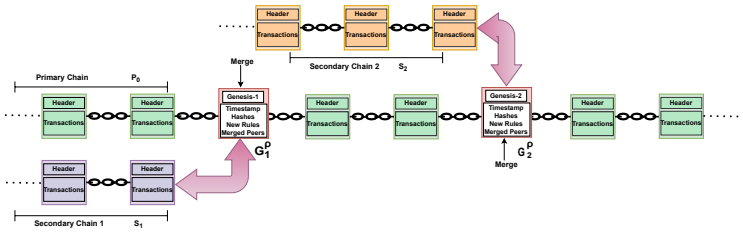


Fig. 2: Framework for homogeneous blockchain networks.

4.1 Working Principal

In this proposed solution for combining similar blockchain networks, we categorize the chains needing consolidation as primary and secondary. The largest, most established network is the primary chain P_0, while the ones that will merge to it are secondary chains $S_1, S_2, ..., S_n$, where n is only limited by network resources. In overextended operations, the P_0 may become a secondary chain during future merger activities as the ecosystem evolves. We introduce specialized genesis blocks. Let G_0^ρ be a genesis block, where ρ is the chain identifier and 0 indicates that it is the first genesis block of ρ. Then, for any merger, a new genesis block will be created as G_i^ρ, where ρ is the chain identifier of P_0, and $i \geq 1$ is incremented for each merger. Hence, a collection of $\mathbb{G} = \{\mathbb{O}, G_1^\rho, G_2^\rho, \ldots, G_n^\rho\}$ will form where ρ will indicate P_0 of that merger, and $\mathbb{O}$ is the set of G_0 for each merged chain. Similarly, $\mathbb{S} = \{S_1, S_2, S_3, \ldots, S_n\}$ is a collection of secondary nodes, where $f : \mathbb{G} \mapsto \mathbb{S}$ $s.t.$ $G_i^\rho \leftrightarrow S_i$.

Figure 2 visually explains the chain merger process, showing the P_0 in green and S_i in different colors. The specialized genesis blocks G_i^ρ depict the position of merge points. Transactions from this point onwards, from all nodes, are then added to a unified chain. This process can be repeated for new networks, ensuring the preservation of histories while enabling harmony between platforms. Before generating each genesis block, consensus mechanisms are aligned across chains, and peer nodes are merged accordingly. This ensures agreement on the state consolidation process and validation rules moving forward. The complete algorithm and individual elements are described next.

4.2 HCC Algorithm

The complete working of HCC is listed in Algorithm 1. Here, CA_1, CA_2, O_1, and O_2 refer to Certificate Authority and Orchestrator/Orderer of P_0 and S_i, respectively. It progresses through peer node combination, governance alignment, and cryptographic fusion phases, ensuring node validation and merging into a joint peer overlay network for coordination. Governance transfer enables unified security for genesis block creation and validation. In the final fusion phase, the last block hashes from both chains are recorded in G_i, along with metadata signatures, binding the histories cryptographically. This process bridges formerly distinct chains, allowing transactions to route across the merged network.

Algorithm 1 HCC Algorithm

Require: P_0, S_i, CA, O
Require: $\mathbb{M} \leftarrow$ all peers of P_0
Require: $\mathbb{N} \leftarrow$ all peers of S_i
Validate $\mathbb{M}$, $\mathbb{N}$
if $\mathbb{M}$ and $\mathbb{M}$ are valid **then**
    $\mathbb{P} = \mathbb{M} \cup \mathbb{N}$ ▷ Peer merge
end if
Validate CA_1, CA_2
Validate O_1, O_2
Transfer Authority: $CA_1 \leftarrow CA_2$
Transfer Authority: $O_1 \leftarrow O_2$
if Transfered **then**
    CA_1 validates $\mathbb{P}$, O_1
    if Validated **then**
        Initialize G_i^ρ
        $b_1 \leftarrow$ most recent block of P_0
        $b_2 \leftarrow$ most recent block of S_i
        $t' \leftarrow$ current time
        prevHash of $P_0 \leftarrow \text{hash}(b_1)$
        prevHash of $S_i \leftarrow \text{hash}(b_2)$
        Append rules for merging
        Assign credentials of P_0 as sign
        O_1 sends G_i^ρ to $\mathbb{P}$ for consensus
        if successful **then**
            Commit G_i^ρ to P_0
            Output: Unified chain P_0
        else
            Rollback
            Output: Commit Failed
            Output: Merge Failed
        end if
    end if
else
    Rollback
    Output: Merge failed
end if

Genesis block: The genesis block ensures identical blockchain rulesets and consensus mechanisms on both networks, forming a fully compatible single network. The genesis blocks contain essential information such as: *Timestamp* indicating the time of the implementation of the migration process. It enables sequential ordering of all blocks created prospectively from the resulting merged chain. *Previous Block Hashes* of the most recent blocks from each of the two chains. This connects the entire history of both chains. By including these previous block hashes, all of the data in both chains are technically still intact and traceable through the genesis block, even after the merge. *Rules* appended for merging. This also includes important mechanism definitions and policy adjustments that will apply to the combined blockchain in the future. *Peers* information is included for authenticated peer credentials from secondary chains to enable permissioning.

Consensus for Merge: The genesis block's consensus algorithm is specialized to unify nodes from both networks onto the same chain. Nodes must accept it as the new starting point. Before each genesis block, consensus mechanisms are synchronized, and peer nodes are merged to ensure agreement, reducing forks and maintaining integrity.

Peers, CA, & Orchestrator Integration: Before merging chains, transferring and validating peers, certificate authorities, and orchestrators is essential. This validation process includes assessing identity bindings, protocol specifications, issuance processes, revocation capabilities, visibility breadth, computational capacities, reliability under load, consensus participation, failure resilience, and examining for prior malicious actions or coding vulnerabilities. These com-

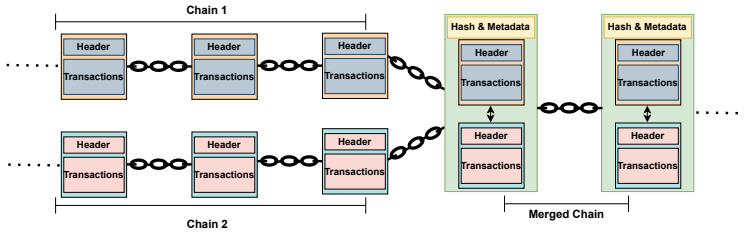


Fig. 3: Framework for heterogeneous blockchain networks.

prehensive reviews and tests ensure that infrastructure and mechanisms are standardized, well-governed, fault-tolerant, and ready for interlinked operation before further consolidation in the merge procedure.

5 HMIA Framework

The proposed approach for Heterogeneous Multi-Chain Architecture (HMIA) introduces a merged chain to connect two existing chains with very low compatibility. Merging blockchain networks with different architectures and consensus protocols can be challenging due to their inherent incompatibility, as each network is designed with its own rules, protocols, and structures.

5.1 Working Principal

HMIA involves the creation of a new merged chain that connects two existing chains by incorporating blocks from each chain into its metablock. In this approach, the merged chain would have a tertiary consensus protocol that allows it to validate its blocks after the consensus completion of both chains. The merged chain does not need to understand all the technical details within each chain, like transaction formats, scripting languages, consensus rules, etc. Only the merged chain hash connects them. Figure 3 illustrates two separate blockchains, Chain 1 in orange and Chain 2 in blue, interconnected through a merged chain (in green). The merged Chain contains special blocks encapsulating the most recent blocks from both chains, interconnected through a Merged chain hash.

5.2 Algorithm

The working of HMIA is defined in Algorithms 2 & 3. The first lists how the merge occurs, and the second lists how subsequent blocks are created in the merged chain M_0. The input consists of two chains S_1 and S_2; the output is the B_0 as a merged block. It contains timestamp information, metarules, hashes, signatures, and the latest blocks of each. S_1 and S_2 continue independently. However, the system is controlled by Orderer/Orchestrator $\overline{O}$, which triggers the individual O_1 and O_2 for respective consensus. Before Commit, $\overline{O}$ performs

Algorithm 2 HMIA Merge Algorithm

Require: Input: S_1, S_2, $\overline{CA}$, $\overline{O}$
$b_1 \leftarrow$ most recent block in S_1
$b_2 \leftarrow$ most recent block in S_2
Interface $\overline{CA} \leftrightarrow \mathrm{CA}_1$, $\overline{CA} \leftrightarrow \mathrm{CA}_2$
Interface $\overline{O} \leftrightarrow \mathrm{O}_1$, $\overline{O} \leftrightarrow \mathrm{O}_2$
Require: $\mathbb{M} \leftarrow$ all peers of S_1
Require: $\mathbb{N} \leftarrow$ all peers of S_2
$\overline{CA}$ validates $\mathbb{M}$, $\mathbb{N}$
if $\mathbb{M}$ and $\mathbb{M}$ are valid **then**
    select $\tau \subset \mathbb{M}$, $\upsilon \subset \mathbb{N}$
    $\mathbb{P} = \tau \cup \upsilon$
    Initialize First block B_0 of M_0
        timestamp $\leftarrow$ current time
        Append b_1, b_2
        Meta info & rules of S_1, S_2
    $\overline{O}$ sends B_0 to $\mathbb{P}$ for consensus
    if successful **then**
        Commit B_0 to M_0
    else
        Rollback
        Output: Commit/Merge Fail
    end if
end if

Algorithm 3 HIMA Block Creation

Require: Input: b_i, b_j
▷ new blocks from S_1, S_2
Trigger consensus of O_1, O_2
if successful consensus **then**
    Initialize B_i of M_0
        timestamp $\leftarrow$ current time
        Append b_i, b_j
        Append Hash of b_{i-1} in b_i
        Append Hash of b_{j-1} in b_j
        Append Hash of B_{i-1}
        Append meta and Hash of B_i
    $\overline{O}$ sends B_i to $\mathbb{P}$ for consensus
    if successful **then**
        Commit B_i to M_0
    else
        Output: Commit Failed
    end if
end if

a consensus using $\mathbb{P}$, a subset of S_1 and S_2 peers. Algorithm (3) outlines the steps for generating subsequent blocks after creating the first block. The specialized consensus is again applied to the new meta-block before committing, along with the necessary metadata and signatures.

Merged Chain Block: The merged chain's first block contains the following fields. *Timestamp:* This initial merged chain block registered the precise date and time when special block formulation was implemented between the separate integrating networks. *Meta Information:* The block stores pertinent descriptive metadata attributes covering hashing algorithms, consensus mechanisms, and architectural properties of the disjoint constituent blockchains pre-integration. *Sub-Blocks:* This particular block contains both chains' latest blocks. The Merged Chain subsequent blocks have an additional *Hash*, which is the hash of the previously merged meta-block and the individual hashes of blocks from S_1 and S_2.

Merged Chain Consensus: The consensus algorithm requires verifying the merged block before including it in the unified ledger. This is done using $\mathbb{P}$, selected from the individual chain peers as shown in Algorithm 2. This consensus can be as simple as validating signatures and as complex as PoS.

6 Evaluation

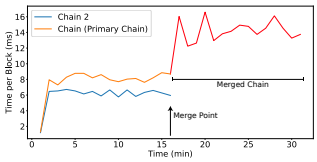
In this section, describe the practical use case for both solutions and then implement and evaluate the algorithms. The testbed is built using multiple systems to model different chains (i5-13500 2.5Ghz with 16GB DDR4 RAM), while the orchestrators runs on a 2.4GHz 10-core Xeon with 16GB DDR4 RAM.

Practical Usecase Analysis: We explore Hyperledger Fabric BC solutions for healthcare data management, focusing on real-world applications. These solutions aim to enhance EHR security, privacy, storage, access, and availability. [22] presents a BC architecture to improve EHR security and privacy, employing OrbitDB with IPFS for storage and Hyperledger Fabric for data hashes and access control. In [20], medical data is managed on a BC using SIFF to encrypt data before storage on Hyperledger Fabric. In [18], a Fabric-based EHR-sharing system is proposed, enhancing data accessibility among healthcare providers. These use cases can be integrated using our proposed HCC, requiring a decision on the storage mechanism before merging, illustrating the merging of business-only BCs with high compatibility.

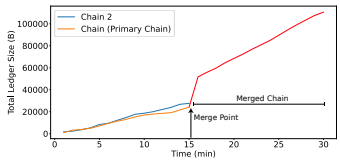
Bitcoin, Litecoin, and Bitcoin Cash can potentially be merged due to their shared features, including block size, consensus algorithm, hash and digital signature algorithms, and smart contract capabilities. However, there is limited compatibility between Hyperledger Fabric and Quorum, despite both using the Raft consensus, private permissions, and ECDSA digital signature algorithm, with differences in block size and smart contract language. BlockMedCare [1], integrates IoT with BC for secure remote patient monitoring. In contrast, [19] presents a framework using Hyperledger Fabric to protect the privacy of EHR. Despite operating in the same domain, these solutions use different BC networks. HMIA allows combining blocks from these networks into a unified chain, which is applicable even when networks are the same but differ in domain and purpose.

HCC Performance: We use Hyperledger Fabric with PBFT consensus and evaluate network performance pre- and post-merge. We compare metrics like consensus time and ledger size to demonstrate the improvements. Figure 4a shows the consensus time per block before and after merging two BC networks (the primary chain and chain 2) over 30 minutes of execution and block time at 1 min. At the merge point (15 min mark), the consensus formation increases as the number of peers and the transaction rate increase for a single orchestrator. However, the change is in ms and acceptable, given the benefits. Figure 4b shows the impact of merging two BC networks on complete ledger size. The total ledger size is added up at the merge point, and then a sharp rise is seen. As the new ledger is a union of all transactions from both networks, hence the sharp rise. However, this is expected and less than the sum of individual systems.

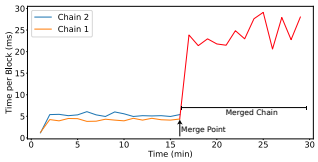
HMIA Performance: For HMIA, we use a hybrid BC architecture with Hyperledger Fabric and Ethereum, each with distinct consensus mechanisms. Hy-



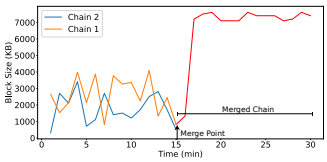
(a) HCC Consensus time



(b) HCC Total ledger size



(c) HMIA Consensus time



(d) HMIA Block size

Fig. 4: Performance Evaluation Before and after merging.

perledger Fabric prioritizes efficiency with PBFT, while Ethereum focuses on efficiency and finality with PoS. We evaluate HMIA using consensus time and block size metrics before and after integration. Figure 4c depicts the consensus time, where the merge occurs at the 15-minute mark. There is a noticeable increase in block finalization time after merging due to the algorithm. Commit has to wait for all consensuses to finish. However, this time can be further reduced by parallelizing the internal consensus and using only validation for the meta block. Figure 4d shows the change in block sizes before and after merging two BC networks. Post-merge depicts a significantly larger block size for the unified chain, incorporating data from both chains. The increase in block size indicates additional storage and data transmission demands post-merge, which is essential for planning multi-chain system infrastructure. Although this seems to have a negative impact, it is less than the cumulative of two separate chains.

7 Conclusion

This paper presented two novel solutions, Homogeneous Chain Consolidation and Heterogeneous Multi-Chain Architecture for merging BC networks. In the first design, compatible BCs based on the criteria defined in this work are merged into a single ledger with multiple Genesis blocks. For those BC systems that are entirely incompatible, we present a meta-block design that enables individual functioning while keeping the ledger as a single meta-chain. These mechanisms are proven functional and beneficial by presenting a use case study of existing work and implementing the solution. The quantitative evaluation proves that the memory and time costs (although increasing) are less than the cumulative of individual chains. In the future, we will expand both solutions to optimize resource usage, evaluate more platforms than listed in this work.

References

1. Azbeg, K., et al.: Blockmedcare: A healthcare system based on iot, BC and IPFS for data management security. Egyptian Informatics Journal **23**(2), 329–343 (2022)
2. Belchior, R., et al.: A survey on blockchain interoperability: Past, present, and future trends. ACM Computing Surveys (CSUR) **54**(8), 1–41 (2021)
3. Biswas, S., et al.: Blockchain for e-health-care systems: Easier said than done. Computer **53**(7), 57–67 (2020)
4. Cordova, D., et al.: Blockgraph: A blockchain for mobile ad hoc networks. In: 2020 4th cyber security in networking conference (CSNet). pp. 1–8. IEEE (2020)
5. De Vries, A.: Cryptocurrencies on the road to sustainability: Ethereum paving the way for bitcoin. Patterns **4**(1) (2023)
6. Deuber, D., et al.: Redactable blockchain in the permissionless setting. In: 2019 IEEE Symposium on Security and Privacy (SP). pp. 124–138. IEEE (2019)
7. Diallo, S.Y., et al.: Understanding interoperability. In: Proceedings of the 2011 emerging M&S applications in industry and academia symposium. pp. 84–91 (2011)
8. Gai, K., et al.: Blockchain meets dag: a blockdag consensus mechanism. In: 20th International Conference on Algorithms and Architectures for Parallel Processing. pp. 110–125. Springer (2020)
9. Garriga, M., et al.: Blockchain and cryptocurrencies: A classification and comparison of architecture drivers. Concurrency and Computation **33**(8) (2021)
10. Houben, R., et al.: Cryptocurrencies and blockchain: Legal context and implications for financial crime, money laundering and tax evasion (2018)
11. Kapengut, E., et al.: An event study of the ethereum transition to proof-of-stake. Commodities **2**(2), 96–110 (2023)
12. KlaytnFoundation: Klaytn finschia mainnet merge `https://govforum.klaytn.foundation/t/kgp-25-klaytn-finschia-mainnet-merge/719`
13. Lin, M.B., et al.: Blockchain mechanism and distributional characteristics of cryptos. arXiv preprint arXiv:2011.13240 (2020)
14. Niya, S.R., et al.: On-chain iot data modification in blockchains. arXiv preprint arXiv:2103.10756 (2021)
15. Pillai, B., et al.: Level of conceptual interoperability model for blockchain based systems. In: 2022 IEEE Crosschain Workshop (ICBC-CROSS). pp. 1–7. IEEE (2022)
16. Ren, K., et al.: Interoperability in blockchain: A survey. IEEE Transactions on Knowledge and Data Engineering (2023)
17. Sebastião, H.M.C.V., et al.: Cryptocurrencies and blockchain. overview and future perspectives. International Journal of Economics and Business Research **21**(3), 305–342 (2021)
18. Tanwar, S., et al.: Blockchain-based electronic healthcare record system for healthcare 4.0 applications. Journal of Information Security and Applications **50** (2020)
19. Thakkar, V., et al.: A privacy-preserving framework using hyperledger fabric for ehr sharing applications. International journal of electrical and computer engineering systems **14**(6), 667–676 (2023)
20. Tian, H., et al.: Medical data management on blockchain with privacy. Journal of medical systems **43**, 1–6 (2019)
21. Xing, X., et al.: A blockchain index structure based on subchain query. Journal of Cloud Computing **10**, 1–11 (2021)
22. Zaabar, B., et al.: Healthblock: A secure blockchain-based healthcare data management system. Computer Networks **200**, 108500 (2021)