International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 13, No.1 (Jan-2023)

http://dx.doi.org/10.12785/ijcds/130115

An In-depth Security and Performance Investigation in Hyperledger Fabric-configured Distributed Computing Systems

Sidharth Quamara[1] **and Awadhesh Kumar Singh**[2]

[1,2]*Department of Computer Engineering, National Institute of Technology Kurukshetra, Kurukshetra-136119, India*

Received 5 Jan. 2022, Revised 27 Sep. 2022, Accepted 12 Jan. 2023, Published 31 Jan. 2023

Abstract: Since its inception as one of the Bitcoin's underpinning technologies, the concept of Blockchain has traversed a long way from being merely a secure distributed ledger meant only for storing cryptocurrencies-based financial transactions to implementing innovative and revolutionary distributed systems for multifarious purposes. One of the contemporary and out-of-the-box Blockchain-based projects, namely Hyperledger, promises to make preeminent use of this technology by promoting cross-industry collaboration in developing Blockchain-based solutions, thus, opening a new chapter in distributed computing. However, attributing to its underlying design, leveraging Hyperledger-Fabric (HF) features still lacks an analysis from the perspective of security risks and efficiency concerns pertaining to real-time distributed computing-based systems and applications. In this regard, we investigate the HF architecture, along with various research endeavours undergone by researchers in recent years to combat its security and performance-related challenges. In light of the identified limitations and bottlenecks, we present our conceptual proposal and feasible insights for improving the efficacy of HF-based systems while not compromising their security.

Keywords: Blockchain, Distributed computing, Hyperledger Fabric architecture, Security and performance issues

1. Introduction

The advent of the Bitcoin project laid the foundation of Blockchain technology that utilizes cryptography and consensus to provide a distributed and tamper-free ledger for storing financial transactions in a decentralized manner, which is a significant milestone in the arena of distributed storage and database technology [1]. Nevertheless, the major limitations associated with Bitcoin included its restricted applicability intended only for the direct transfer of Bitcoin units from one account to another, and its lack of functionality that could provide services like the creation and trading of assets, e.g., Non-Fungible Tokens (NFTs), thus preventing the usage of Blockchain technology to its full potential [2], [3].

To overcome the limitations associated with Bitcoin and other similar systems and harness the features of Blockchain technology to their best, the idea of Ethereum was put forward in 2013 and officially released in 2015 [4]. The proposed system was meant for providing services like allowing users to deploy distributed applications over it as smart contracts. Besides transferring Ether cryptocurrency units between accounts, these distributed applications can flexibly and transparently offer a wide range of services, like trading securities and NFTs. However, despite their advantages over Bitcoin-based systems, Ethereum-based Blockchain systems have two major limitations in their architecture and execution mechanism. In Ethereum-based systems, each smart contract needs to run over all the participant nodes of the concerned Blockchain network, thereby decreasing the transaction throughput [5]. Besides, Blockchain utilized in these systems is permission-less. However, for many enterprises that intend to use Blockchain for specific distributed applications and performance requirements, the permission-less nature of Blockchain is not effective [6], [7]. And in many distributed applications, particularly those meant for financial dealings, knowing the real identity of the participants is crucial for anti-money laundering initiatives. However, in the case of public Blockchain-based systems, revealing the actual identities may not always be safe for privacy preservation.

Some popular projects like Tendermint Chain [8] and Quorum [9] were launched to provide permissioned Blockchain-based systems. However, they inherited one major limitation from previous Blockchain technologies; they also follow the "order-execute-validate" principle, according to which every instruction in the winner block decided after executing the consensus function needs to execute over all the permissioned nodes, thus creating a performance bottleneck. Hence, to provide a broader pathway for promoting communication and collaboration among various heterogeneous Blockchain systems, the project "Hyperledger" was initiated by a group of companies under the umbrella of the

E-mail address: sidharthq.87@gmail.com, aksingh@nitkkr.ac.in

Linux foundation in 2015. A Blockchain framework termed "Fabric" came into existence under the realm of this project as one of its key constituent technologies [10]. The key idea behind the inception of this project was to promote cross-industry collaboration for the development of Blockchain-based distributed ledger technologies and systems, with the main emphasis on the improvement of the efficiency and reliability of these systems for enabling them to support business transactions among major financial, technological, and supply-chain organizations across the globe.

Although a plethora of research proposals have come into the picture in recent years that promise to bring significant enhancements in the overall throughput and reduce the latency of Hyperledger Fabric (HF)-based systems to harness the real potential of Blockchain technology. Still, the performance of these systems lags behind the conventional transaction processing systems. Researchers have proposed to increase the throughput of HF-based systems in terms of Transactions per Second (TPS) (typically 3000-20,000 TPS) by bringing component-level and execution-related changes in Fabric [11], still its throughput is not comparable with the maximum throughput of conventional systems like VISA, which are reported to have shown peak throughput of 56,000 TPS [12]. Hence, the primary motivation behind this paper is to analyze previous research endeavours and propose mechanisms at the conceptual level that may bring further improvements in the efficiency of HF-based systems while not compromising their architectural security.

In-lined with the above-discussed aspects, the main contributions of this paper are as follows:

- We briefly analyze the architecture, including the key components and execution phases involving the interaction among these components of HF-based systems.
- We bring forward and analyze different security vulnerabilities associated with HF-based systems and shed light on various research endeavours in recent years to address the issues due to these vulnerabilities.
- We analyze various research proposals put forward in recent years to improve HF-based systems' performance.
- Finally, we propose our theoretical mechanisms that may reduce the response time of HF-based systems and improve their efficiency while not compromising their security parameters.

In the forthcoming sections of this paper, we discuss the following aspects: Section 2 briefly summarizes the overall architecture of the HF framework, including its key components and phases involved in the applications' execution. Section 3 presents a taxonomy of different security issues affecting HF-based applications. Subsequently, Section 4 highlights various research proposals in recent years to address security issues and performance improvement in HF-based systems. Based on their shortcomings, Section 5 presents our proposal as a first step towards improving the throughput and efficiency of HF-based systems. Finally, Section 6 concludes this paper with perspective research directions.

2. Hyperledger Fabric framework: A technical background

Fabric is primarily a permissioned Blockchain-based distributed computing system, which is both modular and extensible in its design philosophy [13]. The modular aspect of Fabric allows it to decouple the consensus segment from the transaction execution and validation segment, thus authorizing it to select consensus protocol according to the application's requirement. It is the first Blockchain-based system that supports programming in general-purpose programming languages, like C++, Java, and JavaScript. In the following sub-sections, we briefly discuss the components and execution phases as two major aspects of HF-based systems.

A. System components

The constituents of a typical HF-based distributed computing system are as follows, with their roles briefly summarized in Table I:

- **Ordering Service Nodes (OSNs):** OSN nodes receive the endorsed transaction(s), along with their read/write sets from the client(s) of a particular channel that are fed to the atomic broadcast service (generally provided by Apache Kafka). Further, these nodes arrange the transactions received from the atomic broadcast into blocks. Block generation occurs when some prerequisite conditions become true, e.g., when the block has achieved a pre-determined limit of a maximum number of transactions or maximal limit on size (in bytes). The ordering service derived from Apache Kafka is one of the three consensus services currently available.
- **Membership Service Provider (MSP):** MSP is an abstraction that may have different instantiations according to different requirement specifications. The default MSP instantiation in Fabric-based systems provides standard Public-Key Infrastructure (PKI) functionalities for carrying out authentication tasks using digital signatures and generally accommodates commercial Certification Authorities (CAs). In Fabric-based systems, Blockchain networks can be established using two modes: *offline* and *online*. In offline mode, a CA is responsible for generating certificates and distributing them to the nodes. Peers and orderers are registered during the offline mode, while clients are enrolled during an online mode by using CA-issued credentials.
- **Ledger:** Typically, Blockchain ledgers are implemented using a Merkel tree data structure that helps in efficiently storing and searching Blockchain ledger data. In a Fabric-based Blockchain network, it has the following sub-components: 1) a block store and 2) a Peer Transaction Manager (PTM). Ledger block store is implemented using append-only files and retains block containing transactions. It also keeps indices for

TABLE I. Role of different architectural components in Hyperledger Fabric.

Component	Role
Ordering Service Nodes (OSNs)	• Channel re-configuration, when members of the channel broadcast a configuration update transaction. • Access control services provisioning, where they act as trusted entities and may impose restrictions on broadcasting of blocks or transaction endorsements to specific peer(s) according to the requirement.
Membership Service Provider (MSP)	• Authorize the process of identity federation, e.g., when multiple corporations use a Blockchain network. • The membership service system has a dedicated component installed at each node, where it performs: 1) authentication of transactions, 2) verification of the integrity of transactions, 3) signing and validating transaction endorsements, and 4) authentication of other Blockchain-related operations.
Ledger	• Maintaining the ledger and its state on a persistent storage. • Simulation, validation, and ledger-update phase.
Peers	• Recording transaction outcome on a particular local state of the Blockchain ledger. • Sending the outcome, along with the corresponding input/output set back to the client. • Non-endorsing peers participate in validation of transaction blocks transferred to them by ordering sub-system.
Chaincode	• Simulating a particular transaction's proposal, where chaincodes are termed as *application chaincodes*. • Managing overall system's parameters, where chaincodes are termed as *system chaincodes*.

enabling random access blocks or to some specific transactions stored in a block. PTM is responsible for storing the current system state in a versioned key-value database. It maintains one tuple of the type (key k, value v, version ve) for each unique key entry k maintained by any chaincode, having its latest stored value v with the latest version ve.

- **Peers:** Peers are the autonomous computing systems in the Fabric Blockchain network.
- **Chaincode:** Chaincode, also termed a smart contract, is the programming code that can be implemented in any general-purpose programming language, like C++, Java, NodeJs, etc.

A snippet diagram of the flow of messages among various components of a typical HF-based Blockchain system is shown in Figure 1 [13].

B. Phases

Any Fabric-based Blockchain network executes applications (also termed as chaincode in Fabric's terminology) over it, in the following three phases:

- **Execution phase:** In this phase, a client creates a transaction proposal, signs it, and then multi-casts it to a subset of peers (shortlisted according to some specific endorsement policy; sometimes termed as endorsers) in the Blockchain network. After receiving the proposal, each peer simulates it on its local Blockchain state by executing the operation embedded in the proposal payload on the transaction chaincode in a controlled environment or container (generally Docker). Then, each peer in the endorsement list stores the values generated and read as input during simulation in write- and read-set, respectively. Every peer in the shortlisted subset creates a message called an endorsement (that encodes read-set, write-set, endorser's ID, transaction's ID), signs this message, and sends it back to the client from which the corresponding transaction proposal was received. The client keeps receiving the peers' responses until it does not receive endorsements that satisfy the policies in the chaincode. Once sufficient endorsements are received, it assembles a message that contains the transaction's chaincode, meta-data, and a set of endorsements and sends the message to the ordering nodes.
- **Ordering phase:** In this phase, the transactions received by various clients belonging to the same Blockchain channel are totally ordered by the OSNs. OSNs achieve this by atomically broadcasting the endorsements received by the clients by using a consensus algorithm on the sequence of transactions. At a high-level interface, different services are provided by an ordering service. It is essential to understand that OSNs do not have any state of the Blockchain and do not participate in the execution and validation of the transaction. Because of this design philosophy, consensus functionality is modular in Fabric and can be easily updated or changed according to the requirements.
- **Validation phase:** It comprises the following three steps: 1) In the first step, all the transactions within the concerned block are evaluated in parallel concerning the endorsement policies. This is carried out by Validation System Chaincode (VSCC), part of the Blockchain configuration library; 2) In the second step, read-write conflicts of the transactions are analyzed; and 3) The ledger update phase executes in the end in which the block is appended into the local state of the Blockchain ledger. The state of the Blockchain is updated, and validity check results in the first two steps are also retained.

3. Taxonomy of security vulnerabilities in Hyperledger Fabric-based systems

HF being a distributed ledger platform employing autonomous computing agents (e.g., clients, peers, endorsers, MSP), communication network and programming code running over these components as chaincode, introduce their own set of security-related challenges. On this basis, security vulnerabilities pertaining to HF-based Blockchain systems can be broadly classified into Fabric chaincode-

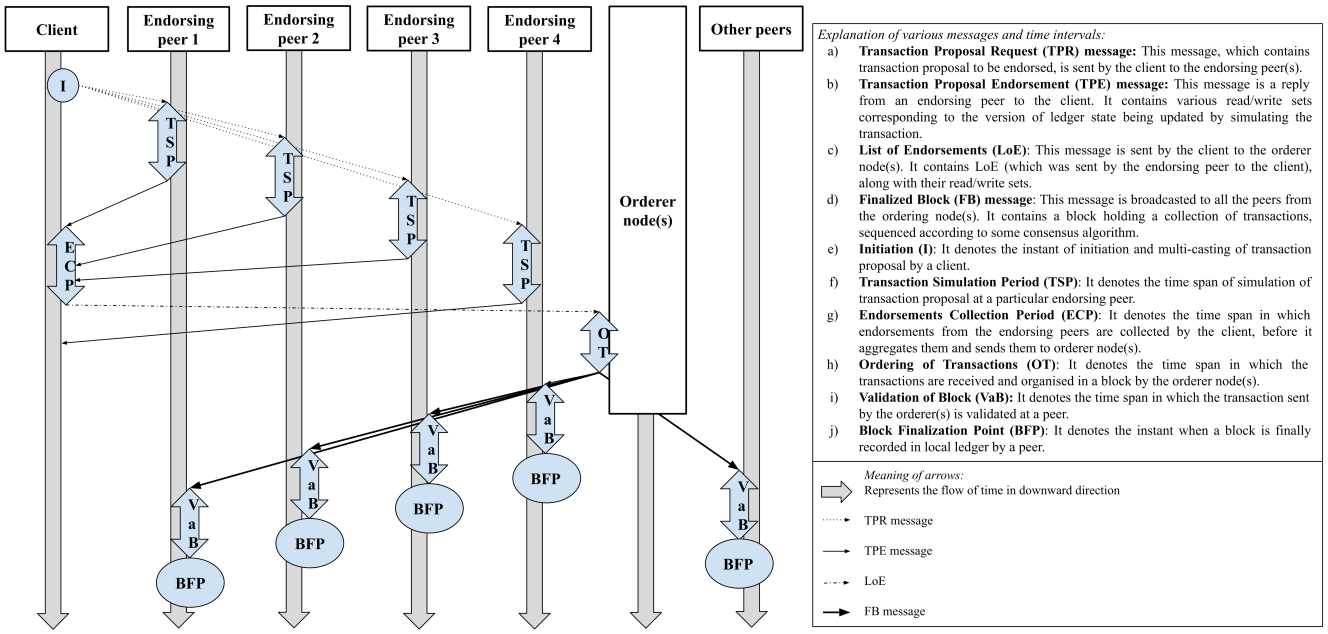

Figure 1. A general message-flow diagram of Hyperledger Fabric explaining the necessary messages and time spans.

related vulnerabilities, and HF Blockchain communication network and nodes-related vulnerabilities, as detailed in the forthcoming sub-sections.

A. *Fabric chaincode-related vulnerabilities*

Since Fabric chaincodes are written in general-purpose programming languages (refer to Section 2-A), all the anomalies that may affect routine programs due to the loopholes inherent in programming languages or practices may also influence Fabric chaincodes. Some of these major security vulnerabilities are mentioned as follows:

- **Vulnerability because of non-determinism in programming or source code of fabric chaincode:** For transaction outcomes or endorsements submitted by the endorsing peers to be accepted by the client, the main requirements are that these should be in accordance with the pre-determined endorsement policies and the endorsements submitted by different peers should be the same [14]. To this end, there should be no non-deterministic instruction in the proposed transaction chaincode. Otherwise, different peers will generate disparate outcomes despite identical local Blockchain states, affecting the overall execution phase. Some possible causes of non-determinism are: 1) random key generation and 2) usage of system timestamp or global variables not stored in the Blockchain ledger.
- **Vulnerability because of log injection:** Logs are databases intended for storing information that can be used for the process of debugging, data collection, and performance optimization. A significant volume of work and important decisions are taken in the context of various applications based on data stored in these logs. Because of the important role a log plays, it is a choice of following security attacks and exploitation by the attackers: 1) fabrication of log messages, 2) corruption of log data, thus averting its normal processing, 3) creation of emulators and dedicated applications that provide the attacker with a chance to have a glimpse on the processed logs, which may lead to the previous two attacks [15].
- **Vulnerability because of concurrent execution:** If multiple transactions are executing concurrently under high load, then an accidental change in a particular key version may cause key collisions [16]. Subsequently, this may lead to double-spending attacks. For example, on successful completion of the endorsing phase by a transaction, an accidental modification in its key version before reaching the validation phase may not only lead to a program error but also permit the inclusion of other transactions in the ledger, possibly causing double-spending.
- **Vulnerability because of chaincode sandboxing:** Although the chaincode of the proposed transaction sent by the client executes in a controlled environment or container (e.g., Docker) at every peer in the shortlisted subset, still it can be exploited by an outside attacker for carrying out malicious activities. The use of a controlled environment imposes some constraints on the chaincode's execution behavior, albeit chaincode is capable of conducting the following activities: 1) installation of arbitrary software that acquires the container environment; e.g., security applications like Nmap, 2) performing port scan activities on public or private networks, which are visible to the nodes, 3) exploiting discovered vulnerable hosts, 4) receiving

commands and transmitting back the results to a remote malicious server, and 5) continuing self-execution for an indefinitely long period. Remote Access Trojan (RAT)-based malware can be created using these capabilities, thus leading to the generation of a foothold in a corporate setting meant for scanning and attacking other systems [15].

[**NB:** The vulnerabilities mentioned above are associated with HF-based systems. Indeed, exposure of the underlying consensus mechanisms may impact the usage of Blockchain in these systems. However, more work is needed in this direction.]

B. Hyperledger Fabric blockchain communication network and nodes-related vulnerabilities

Since the autonomous computing nodes involved in HF platform communicate with each other via messages across the communication networks, there may be attempts to exploit network-related loopholes and vulnerabilities to carry out security breaches and unauthorized accesses on these nodes. Some major vulnerabilities associated with this aspect are as follows:

- **Vulnerability because of compromised Membership Service Providers:** MSP is a crucial component of an HF-based system since it is responsible for registering new members, maintaining the identities of the members, and dealing with access rights of all system nodes. Therefore, if the integrity of MSP is compromised, all the administrative functionalities like adding new identities into the Blockchain network and removing some members from the network, as well as granting and revoking access rights, will come under the control of the attacker. Thus, an attacker can cause damage to the system and may also carry out attacks, such as double-spending attacks, invalid identification attacks, etc. [17].
- **Vulnerability because of endorsers whose identities have been revealed:** In HF-based systems, endorser nodes are responsible for simulating transaction proposals sent by the clients and validating them after receiving them from orderer nodes. Hence, when the identities of endorsers are revealed, the following attacks may be conducted [17]: 1) some specific endorsers may be subjected to Denial of Service (DoS) attacks either for causing the degradation of the system's performance or preventing the inclusion of a particular transaction into the Blockchain, 2) each endorser node has its independent and autonomous course of action to determine the validity of transactions. Therefore, if the identities of the endorsers are revealed, it may cause conflicts among endorsing peers and compromise their ability to act independently, and 3) a particular endorsing peer may leak confidential information about other endorsing peers to an outside adversary, which may lead to further attacks on endorsers [18].
- **Vulnerability because of underlying consensus protocols:** Consensus process is one of the key constituents of any Blockchain technology-based ecosystem that assists applications in providing secure, reliable, and decentralized services. Because of their significant role in these applications, there have been many attempts to find loopholes in different consensus protocols and various ways to exploit them for any unauthorized and malicious task. Some of the major security vulnerabilities associated with consensus protocols include, but are not limited to single-point-of-failure, system crash due to influence of malicious ordering nodes, etc. [19].

4. Research endeavours for combating security and performance issues with Hyperledger Fabric-based systems

In the following sub-sections, we mention some important research endeavours carried out in recent years to address the security- and performance-related challenges in HF-based systems.

A. Combating security issues with Hyperledger Fabric-based systems

A brief summary of research contributions addressing security issues in HF-based systems is provided in Table II and detailed as follows:

1) Dealing with fabric chaincode-related security issues

Fabric chaincodes are one of the core pillars of an HF-based system. Since they encapsulate the computation part for executing contracts and subsequent transactions, their security is of paramount concern. The lack of transaction privacy over Blockchain is a significant hurdle in the wide-scale adoption of decentralized smart contracts. Stressing this concern, the authors in [20] proposed a decentralized smart contract-based system called "Hawk" to maintain contractual security and on-chain privacy of users' transactions. Likewise, concerning the criticality of trustworthy data feeds in developing decentralized Blockchain smart contracts, the authors in [21] presented an authenticated data feed system termed "Town Crier". It is centered on Intel's Software Guard Extensions (SGX) technology, in combination with the smart contract's front end. It supports numerous security features, including confidentiality, the transmission of private data requests, and controlled access to online data resources.

A formal analysis and verification mechanism for HF chaincodes is developed in [22] by extending "KeY" [23], a software developed for proving the correctness of a Java application, by adding Java Modeling Language (JML) specifications of Fabric Application Programming Interface (API) into KeY. The authors of the concerned work developed techniques to reason about serialization and persistence of the objects stored over the ledger. For correctness verification and fairness validation of a smart contract, the authors in [24] proposed an automated framework termed "Zeus", utilizing abstraction interpretation and symbolic

model checking. This framework takes high-level PL-based smart contracts or chaincodes as input and generates correctness and fairness criteria in eXtensible Access Control Markup Language (XACML)-styled templates.

2) Dealing with fabric blockchain network-related security issues

Due to the visibility of transactions across the entire Blockchain network, sensitive user information, in some instances, is subjected to a lack of privacy. In this regard, the authors in [25] presented a scalable and flexible architecture for an HF-based MSP to make peer communications more secure and reliable. The authors concluded that integrating SGX in the MSP can secure it across all the phases (e.g., registration of the nodes, transaction's signature verification) of its working and deter an attacker from compromising it. Regarding identity management of users, the authors in [26] presented a method based on "Black Ridge" technology for Cloud-based Blockchain applications. The authors proposed that if network segmentation and traffic separation are done based on user identities, multiple users can share the identical Blockchain with reduced risk of Distributed Denial of Service (DDoS) attacks over the system.

Two other security limitations with regards to HF include 1) DoS attacks that may be carried out on endorsers whose actual identities are revealed, and 2) a particular member of the Fabric network after coming under the influence of an external attacker because of a wormhole attack, may disclose sensitive information of Blockchain network. For dealing with the first problem, the authors in [27] proposed two approaches. The first approach uses a random verifiable function for randomly selecting endorsers. In their second approach, the identity of the endorsers is kept anonymous using pseudo-identity, i.e., endorsers use pseudonyms while endorsing transactions. For dealing with the second problem, the authors again proposed two approaches. In their first approach, a group signature algorithm anonymizes a peer's identity. Their second approach used bi-linear pairing to achieve the receiver's anonymity.

In [28], the authors addressed the issue of biased voting by endorsers in Fabric-based Blockchain networks by proposing a "Fabric's Constant-Sized Linkable Ring Signature (FCsLRS)" in which the actual identity of endorsers participating in the endorsement process is kept hidden. The proposed scheme works on a threshold endorsement policy according to which at least k out of n (where $1 \leq k \leq n$) endorsers need to endorse a transaction without explicitly revealing their identities, where k is the threshold, and n represents the total number of members in the endorsers set.

B. Research endeavours for combating performance issues with Hyperledger Fabric-based systems

One of the major drawbacks associated with HF and other similar Blockchain-based technologies is that their transaction throughput is comparatively lower compared to conventional Relational Database (RDBMS)-based transaction processing systems. Some significant research endeavours have been observed in recent years for understanding the performance bottlenecks in HF-based systems, along with various mechanisms for improving their efficiency, as discussed in the following sub-sections. Table III summarizes these research contributions.

1) Improving efficiency by re-designing validation, verification, and committing phases

A thorough study is conducted by [29] for understanding variations in transaction throughput and latency in the HF platform by bringing changes in the following system configuration parameters: 1) chosen endorsement policy, 2) size of the block containing transactions, 3) channel-related aspects and parameters, 4) resource allocation policies, and 5) type of state database. Smaller block size is always recommended for achieving a lower transaction latency in cases where the rate of arrival of transactions is expected to be lower than a particular saturation threshold. For attaining lower transaction latency and higher throughput, heterogeneous peers should be avoided, and at least one validation Central Processing Unit (CPU) should be allocated per channel.

A study was carried out in [30] on the performance of the validation phase in HF, wherein it is observed that the validation of transactions is significantly slower with the CouchDB database than with the LevelDB database. The authors proposed re-designing the validation phase so that validation of transactions and reads from the state database could be carried out in parallel and executing ledger and state database writes in parallel. The authors in [31] proposed to remove lock contentions before accessing the cache, thus improving MSP caching. As a result, the throughput is reported to increase to 25,000 TPS. A critical performance-related study has been conducted on HF 1.2, and an observation is made that although some earlier proposals [29], [30], [31] reported substantial improvements; however, their main emphasis was on caching-based upgrades. After the critical observations from their study [11], researchers have recommended re-architecting the Fabric-based system by incorporating several optimizations derived using common system design principles. The proposed changes have been reported to bring performance gains in end-to-end transaction throughput by a multiple of 7 (approximately 3000-20,000 TPS) while simultaneously reducing the block latency.

2) Improving efficiency by analyzing mechanisms for query retrieval execution process on state database and Blockchain ledger in Hyperledger Fabric

The authors in [32] observed the importance of various versions of a particular key in a typical HF-based system as a source of temporal data. They proposed that this data analytics can generate valuable business insights for various use cases (e.g., in supply chain management). In HF-based systems, current state data is stored on a database, while

TABLE II. A brief summary of some major research endeavours for addressing security issues in Hyperledger Fabric-based systems [✓/×: Aspect is/not in the scope of the contribution].

Ref.	Scope				Empirical verification	Use case(s)
	Key contribution	Target security issue(s)	Placement in HF-deployment	Salient feature(s)		
[20]	Decentralized smart contract system "Hawk" for building privacy-preserving smart contracts	Privacy exposure of transaction-related data (e.g., transaction values, public keys) on Blockchain	Between users and Blockchain	• On-chain privacy • Contractual security	✓	• Sealed second-price auction • Crowdfunding • Lottery game • Swap financial instrument
[21]	Authenticated data feed system "Town Crier (TC)"	Lack of trustworthy data feeds	Between websites and smart contracts	• Data feed authenticity • Requests' confidentiality	✓	• Financial derivative • Flight insurance • Steam marketplace
[22]	Deductive formal verification approach for HF smart contracts	Security exploits due to immutability and exposure in public setting	Developer and user-end	• Serialization reasoning • Object persistence	×	Rock-Paper-Scissors game
[24]	Automatic formal verification and fairness validation framework "ZEUS" for smart contracts	Immutability and bug patching difficulties in smart contracts	Developer-end	• Correctness bugs handling • Execution semantics encoding	✓	Simple Dice
[25]	Privacy preserving and trusted membership service architecture in distributed ledger	Visibility of transaction-related sensitive information across network nodes	Protocol-level	• QUOTE authenticity • Selective attribute disclosure • Auditing of auditor	×	-
[26]	User identity management method for Cloud-based Blockchain applications	Distributed Denial-of-Service (DDoS) attacks	Between Blockchain client and Fabric/server-side application	• Network segmentation • Traffic seperation	✓	-
[27]	Anonymity mechanisms to randomize and anonymize endorsers	Denial-of-Service (DoS) attacks on endorsers and wormhole attacks on the network	Fabric and channel-level	• Endorser randomization using Verifiable Random Function (VRF) • Endorser anonymization using pseudonyms	✓	-
[28]	Anonymous endorsement system and ring signature scheme	Conflicts in transaction endorsement due to identity disclosure	Fabric-level	• Threshold endorsement policy for transaction approval • Prevents multiple signatures by same endorser	✓	-

historical data is distributively stored on a set of blocks across the file system. Because of this approach, analytics of history data becomes quite cumbersome and inefficient. To deal with this, the authors proposed two mechanisms for improving the query execution on history data. In the first mechanism, they created separate temporal indexes on history data. They stored these indexes in the history database as key-value pairs for fast retrieval of the history database. In the second mechanism, instead of creating additional key-value pairs for index data, they proposed to transform the (*key*, *value*) tuple by embedding indexing data with different key-value pairs and store this transformed tuple (*index*, *key*, *value*) on history database, thus improving the query retrieval process.

Transactions with concurrency-related conflicts are one of the major performance bottlenecks in HF-based Blockchain systems because such transactions are detected in the final stages of execution, which causes inefficient transaction processing. To address this concern, the authors in [33] proposed a novel approach termed "Locking Mechanism and Ledger Storage (LMLS)". This approach consists of two parts: 1) in the first part, a locking mechanism is used to discover conflicting transactions in the initial stages of transaction flow. If a particular transaction proposal is initiated, first, it is tried to find out whether its input key is locked or not, thus determining the possible conflict with some earlier transaction, and 2) in the second part, according to the locking technique used in the first part, database indexes of transactions generating conflicts are modified and temporally stored, thus improving the efficiency of transaction processing.

A benchmark termed "Hyperledger Fabric GoLevelDB (HLF-GLDB)" for performance characterization of database accesses in HF-based systems is put forward in [34]. It is proposed that by using HLF-GLDB, database accesses-related performance parameters can be easily investigated without having the need for building up a complete HF environment. Using their proposed framework, the authors made the following significant observations: 1) data compression process is a major source of a performance bottleneck, specifically during the database access phase. Performance gains by 54% are reported by disabling the data compression process, 2) database size is a significant factor affecting HF's performance significantly. Degradation in performance by 25% is declared when the database size is increased by 4 times, and 3) accessing a single large value is comparatively faster than multiple component values that are split from the corresponding single large value.

3) Improving efficiency by re-designing block propagation phase

In a Blockchain-based distributed system, block propagation delay is one of the leading causes that significantly affect the TPS throughput parameter, in turn affecting the scalability. In this regard, some major research proposals put forward in recent years to bring improvements in the block propagation phase are discussed as follows.

The authors in [35] proposed a new broadcast overlay architecture called "Kadcast" for Blockchain networks. Kadcast helps achieve a better broadcast operation by utilizing tunable redundancy and overhead. It uses a structured overlay topology for delegating broadcast responsibilities to sub-trees with decreasing height. The authors have proposed incorporating Forward Error Correction (FEC) in their proposed approach, for achieving a highly reliable and resilient delivery of service, even in scenarios of packet loss and adversarial node failures.

In [36], two major limitations with the *infect-and-die* push-based block propagation approach of HF are identified: 1) high communication overhead and 2) low probability of transferring blocks to all peer nodes. To overcome these limitations, the authors proposed the "infect-upon-contagion push" approach for the architecture of HF-based

systems. In this approach, every peer must disseminate blocks to others as soon as they receive them in each round. The main advantage of this approach over the infect-forever model is that it promotes load balancing, as the entire burden does not fall onto the peer initiating the rumor. In this approach, a counter r (initialized to 0) is associated with each block b. When a block b with counter $r = i$ is received by a peer node p for the first time, r is incremented by 1 for i to $i + 1$ by the peer, and then, block b is transferred to a sample of other peer nodes chosen at random. This dissemination process stops when the counter associated with the block being disseminated becomes equal to a pre-determined Time-to-Live (TTL) parameter value. The notion of rounds is not required in this setting. The authors also removed the $tpush$ = 10ms timer associated with data blocks, which is present in the original Fabric architecture, to ensure unbiased randomness. They observed that as the TTL parameter increases, most of the peer nodes are already informed at the final stages of the dissemination phase, and they keep forwarding the same blocks to each other.

The authors in [37] proposed a quick block dissemination system for HF-based Blockchain systems based upon Device-to-Device (D2D)-based 5G networks. The proposed architecture utilizes locality-aware D2D networking for achieving block dissemination at faster rates. Authors have observed that in the conventional Gossip protocol, there is a gradual increase in transactions with network delay because of block propagation delay. However, authors have claimed that there is significantly less variation in transaction confirmation in their proposed system because of the incorporation of a dynamically organizing overlay structure that helps achieve fast block dissemination.

4) Improving efficiency by re-designing the consensus approaches

The initial version of HF was implemented without using a Byzantine Fault Tolerance (BFT)-based consensus service. To overcome this, the authors in [38] proposed a consensus approach based on BFT-SMART state machine replication/consensus library for HF. Using this technique, throughput up to 10,000 TPS can be attained, and a transaction can be irrevocably recorded in the Blockchain in half a second despite consensus nodes being scattered over different continents. The authors in [39] analyzed that although asynchronous BFT protocols [40], [41], [42], [43] are appropriate approaches for deploying Wide-Area Network (WAN)-based permissioned Blockchain because of their robustness against timing and DoS attacks, yet some limitations tend to discourage the use of these protocols in practical distributed Blockchain-based applications. One major limitation is that protocol Honeybadger [43] is observed to have higher latency than partial synchronous protocols, e.g., in [44] because it is based on expensive threshold cryptographic schemes [45], [46]. To address the challenges mentioned above, the authors proposed "BEAT" as a collection of practical asynchronous byzantine protocols.

The authors in [47] observed that although leader or coordinator-based schemes are effective in bringing consensus in asynchronous Blockchain-based systems, their main limitation is that they work on the assumption of partial or eventual synchrony, and the leader will never fail. Its integrity will be intact, which cannot always be guaranteed, and a faulty leader can bring down the consensus algorithm's performance and impose its value on other members [48], [44], [49], [50], [51], [52], [53]. To deal with this limitation of leader-based consensus approaches, the authors proposed a novel scheme termed "Democratic Byzantine Fault Tolerance (DBFT)". They presented a solution for the binary consensus with the assistance of a weak coordinator that eliminates the need for either signatures or randomization. A weak coordinator used here that cannot impose its value has been used. In their proposed approach, non-faulty processes can quickly agree on a value without the coordinator's help. The role of a weak coordinator here is only to assist in the successful termination of the consensus algorithm, if non-faulty processes have been analyzed that the values proposed by them might all be decided.

A virtual voting-based consensus approach is proposed in [54] to reduce the overall message communication overhead in the network compared to the previous consensus approaches. In this approach, all the nodes that want to participate in the consensus process execute a virtual voting procedure locally on a data structure called *Hashgraph* to determine the temporal order of various events in the system for reaching consensus on the sequence of those events. Initially, different participant nodes may have different versions of the Hashgraph data structure in the earlier phases; however, with time and with the help of regular gossiping of its latest versions, all nodes will eventually have the same version of Hashgraph. A typical example diagrammatic representation of Hashgraph is shown in Figure 2. The two events on different nodes connected by an arrow are the events that also include message sending and receiving operations besides other operations.

The authors in [55] thoroughly analyzed the Hashgraph consensus mechanism and observed that a great amount of effort is required to take a snapshot for the Hashgraph to release the memory because of the complexity of the confirmation protocol. Moreover, at least three rounds of voting are required to reach a consensus. To overcome these limitations, the authors in [54] devised a modified and highly efficient Directed Acyclic Graph (DAG) scheme derived from the Hashgraph consensus approach, termed as *Jointgraph*. It incorporates a weak supervisor node in the architecture of Hashgraph, because of which only one round of voting is required. Every member node maintains a copy of the Jointgraph and carries out virtual voting on its copy, followed by the analysis of what vote the others would have given to it. Members do not require to transmit their votes, so zero bandwidth is needed other than for simply gossiping of events. Supervisor nodes also keep track of the behaviors and activities of the ordinary nodes, try to

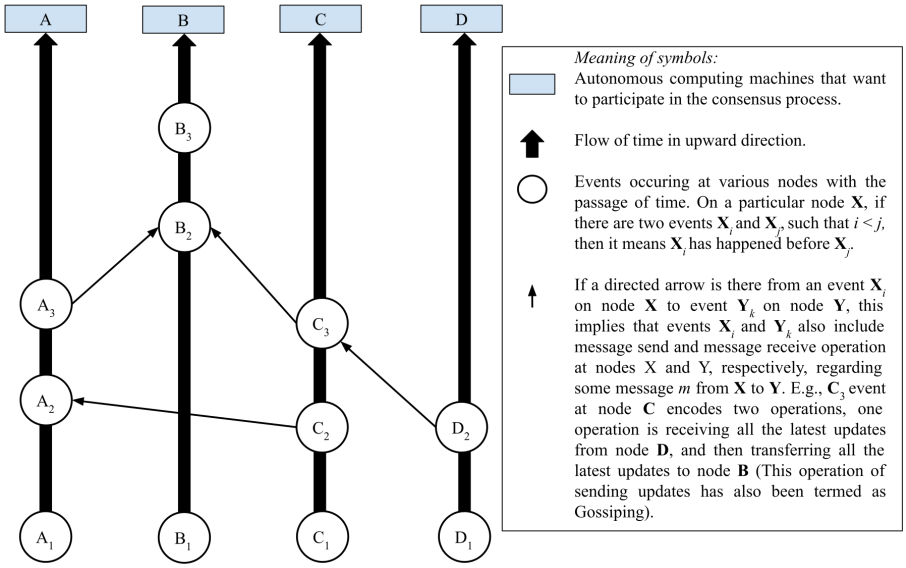

Figure 2. A general diagrammatic representation of Hashgraph data structure.

identify any malicious behavior pattern, and take a snapshot of the system that routinely replaces the ordinary nodes if required. If it is detected by other member nodes that the supervisory node has been hacked down and is not functioning properly, then in that case, the consensus mechanism can shift to the Hashgraph approach. By repeatedly simulating the Jointgraph consensus mechanism on different scenarios, it is observed to be much less effective in latency in comparison to Hashgraph.

5. Perspective research proposal

It can be observed that although cryptocurrencies were created by emphasizing the decentralization notion, because of practical limitations, e.g., the high amount of costs associated with dedicated computing and storage systems for mining and their associated maintenance and running costs, make them only accessible to less than 20% mining coalitions [56]. As a result, a byzantine quorum-based system with a size of 20 can attain better decentralization at a much lesser price than a Dedicated Proof-of-Work (DPoW) mining-based system. This has inspired the development of Scalable Byzantine Fault Tolerant (SBFT) replication systems that can scale to several replicas and be optimized for global WAN [57]. The proposed system is evaluated in a world-scale deployment having 200 replicas, which can withstand up to $f = 64$ Byzantine failures, where f denotes the number of faulty replicas in the system. Researchers put forward results of various experiments to bolster their claim that SBFT-based systems can provide approximately 2 times better throughput and 1.5 times better latency than Practical Byzantine Fault Tolerant (PBFT)-based protocols [58].

One possible solution for increasing the throughput within a Blockchain channel is that when every client associated with a particular channel has achieved minimum credibility and financial stake in the Blockchain system, there should be the provisioning of the following:

1) allowing clients to submit their proposals for relaxing the consensus process for next k consecutive transactions, where $k \leq n$ (n being decided by the policy of the overall network and concerned Blockchain channel), and

2) choice regarding which consensus mechanism should be used for $k + 1^{th}$ transaction.

These proposals can be piggybacked with i^{th} transaction, and only that client's proposal will be accepted, whose transaction will come at the top after the current consensus execution. Hence, relaxing the consensus at regular intervals may drastically improve the overall network throughput. If a malicious client tries to take some undue advantage of the above-mentioned relaxation, the credibility of all the clients will be deducted. And suppose their credibility falls below a certain threshold. In that case, they will have to achieve that credibility again to be able to request relaxation in consensus and their choice for consensus in the next transaction.

Another possible way to decrease the response time and increase the throughput is that when a particular client has achieved some minimum credibility and financial stake in the system, it should be allowed to have its final transaction endorsement result be stored in a temporary stub (being protected by Intel's SGX) at all the peers, after having being simulated and validated by a sufficient number of endorsing peers such that all the references to the concerned transaction be directed to the stub until the block containing this transaction is created and broadcasted by the orderer sub-system. To avail the service mentioned above, the client

TABLE III. A brief summary of some major research endeavours for addressing performance issues in Hyperledger Fabric-based systems [✓/×: Aspect is/not in the scope of the contribution].

Ref.	Scope				Empirical verification	Evaluation metrics
	Key contribution(s)	Target performance issue(s)	Placement	Salient feature(s)		
[11]	Architectural optimizations in HF	Scalability issues in BFT consensus algorithm	Developer end	Independent optimizations concerning caching, input/output, parallelism, and data access	✓	Latency, Throughput, Execution time
[29]	Empirical study for HF performance characterization	Verification of endorsement policy, performance parameter configuration, sequential policy validation of block transactions, state validation and commit	HF component-level	Throughput enhancement	✓	Throughput, Latency, CPU utilization, Endorsement timeout, Lock holding duration
[30]	HF validation phase architecture	Absence of full utilization of computational resources in sequential operations, performance degradation in static information access from state databases	Peer nodes	• Throughput enhancement • Parallel database read/write operations	✓	Throughput, Latency
[31]	Profile-based approach for layered bottleneck detection	Performance model and layered bottleneck measurement from system execution profile	Software nodes	• Limited runtime overhead in profile collection	✓	Throughput, Resource utilization
[32]	Model for processing temporal queries on HF	Limited Application Programming Interface (API) for temporal data access from the file-system, lack of support of temporal indexes	Fabric-level	• Performance enhancement in temporal query handling • Potential generalization to analytical queries	✓	Join time, Index construction time, Data ingestion time, State access cost
[33]	LMLS approach for HF performance optimization	Limited transaction throughput for transactions with concurrency conflicts	Transaction flows	• Discovery of conflicting transactions • Ledger storage optimization	✓	Throughput, Transaction efficiency
[34]	HF performance characterization	Database accesses	HF database systems	• Facilitates performance investigation without the need of complete HF environment setup	✓	TPS, Execution time breakdown
[35]	Structured broadcast approach for Blockchain networks	Message complexity and overhead in broadcast, delayed or inefficient block propagation	Protocol-level	• Tunable overhead • Reliable FEC	✓	Broadcast reliability, propagation delay, network coverage, stale rate
[36]	Gossip design in Fabric	Broadcast performance and scalability	Protocol-level	• Faster block dissemination • Reduction in invalidated transactions for concurrent applications	✓	Propagation time, latency, bandwidth consumption
[37]	Block transport system for 5G-enabled HF Blockchain	Block dissemination delay, scalability, ledger information inconsistencies	Protocol-level	• Status tracking of mobile peers • Faster block propagation	✓	Transaction confirmation latency, Block dissemination and reception delay, TPS throughput
[38]	BFT Ordering Service for HF platform	Lack of BFT ordering service	HF component-level	• Increase in transaction processing throughput • Irrevocable transaction writing across different continents	✓	Throughput, Latency
[39]	Practical BFT protocols set BEAT for fully asynchronous environments	High latency and low throughput in low computational power scenarios	Protocol-level	• Modular design • Extensible	✓	Throughput, Latency
[47]	Leaderless Byzantine consensus algorithm DBFT	Deterministic consensus resolution	Protocol-level	• Uses weak coordinator with no proposal imposition • Time and resilience optimal • Does not require signatures	×	-
[54]	Swirlds hashgraph consensus algorithm for replicated state machines	Increase in network traffic while ordering transactions	Protocol-level	• Bandwidth efficient due to virtual voting • Fair and fast	×	-
[55]	A DAG-based consensus algorithm for consortium blockchains	Communication overhead, scalability, dynamic user participation	Protocol-level	• Member behaviour monitoring by a supervisor • Improved throughput and latency	✓	Throughput, Consensus time

will need to pay pre-determined fees to the system for a specific transaction.

The diagrammatic representation of the flow of messages in the proposed approach that may reduce the overall response time is shown in Figure 3.

6. Conclusion and future research directions

In this paper, we presented and analyzed the key aspects associated with a typical HF-based Blockchain system. Moreover, we shed light on major security-related issues that may degrade the services of these systems, along with various research proposals presented by researchers in recent years to combat these issues. Besides, we outlined some recent proposals for improving the efficiency and throughput of HF-based systems. In the end, we briefly presented our proposal comprising conceptual mechanisms to improve the overall efficacy of these systems.

We have proposed theoretical approaches that can improve the performance of HF and reduce the response time for a particular transaction if these are implemented carefully by taking cognizance of the security requirements of the actual system. Therefore, the developers and administrators of the system have to identify the suitable parameters required for determining the credibility requirements of the clients of these systems through some policy specifications who want to avail benefits by using our theoretical approaches. The mentioned aspects need to be evaluated regarding a practical implementation in a given deployment scenario.

Another major research problem that remains to be explored is to increase the fault tolerance of Blockchain channels in an HF-based network. Suppose some nodes in a particular HF channel are not working correctly because of malfunctioning. In that case, there should be provisioning to offload some transactions to peers of any other Blockchain channel of the same HF network without compromising the security and confidentiality of the offloaded transactions and maintaining the current efficiency of the system. Although the advent of Trusted Execution Environments (TEE) (e.g., Intel's SGX) assisted in popularizing offloading certain computing tasks [59], they have their own security-related limitations that may be exploited to carry out relay attacks, and subsequent physical and side-channel attacks [60]. Despite the proposals put forward by researchers to counter these attacks, these require incorporating dedicated trusted embedded devices into the target hosts that may increase the overall system costs and need prior approval of the target hosts [60], [61]. Therefore, an important aspect yet to be worked upon is to seek solutions to address the security vulnerabilities of TEE without needing to integrate additional dedicated and costly hardware devices into the target systems so that the overall system costs should be

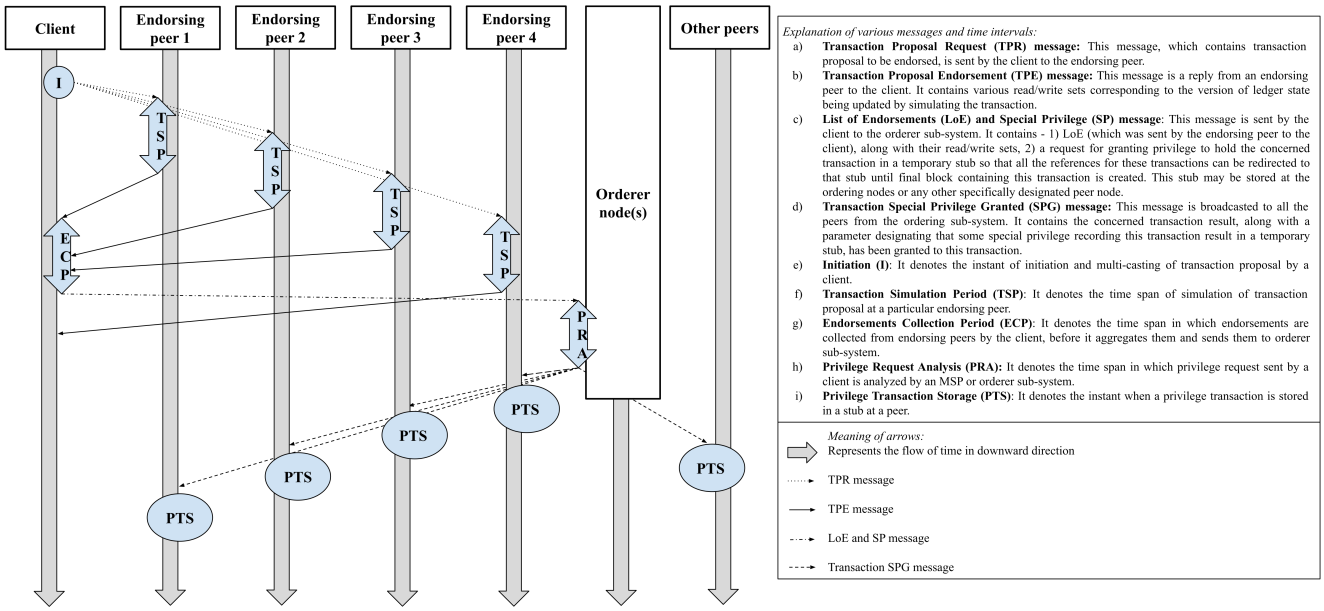

Figure 3. A proposal to record a particular transaction in a temporary stub at orderer nodes (after being simulated and validated by sufficient number of endorsing peers) until the formal block recording the transaction is created by the orderer nodes.

within the budget of clients and corporations.

References

[1] S. Nakamoto, "Re: Bitcoin p2p e-cash paper," *The Cryptography Mailing List*, 2008.

[2] I. Alqassem and D. Svetinovic, "Towards reference architecture for cryptocurrencies: Bitcoin architectural analysis," in *2014 IEEE International Conference on Internet of Things (iThings), and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom)*. IEEE, 2014, pp. 436–443.

[3] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten, "Sok: Research perspectives and challenges for bitcoin and cryptocurrencies," in *2015 IEEE symposium on security and privacy*. IEEE, 2015, pp. 104–121.

[4] V. Buterin *et al.*, "Ethereum white paper," *GitHub repository*, vol. 1, pp. 22–23, 2013.

[5] M. Vukolić, "Rethinking permissioned blockchains," in *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*, 2017, pp. 3–7.

[6] A. Lohachab, S. Garg, B. H. Kang, and M. B. Amin, "Performance evaluation of hyperledger fabric-enabled framework for pervasive peer-to-peer energy trading in smart cyber–physical systems," *Future Generation Computer Systems*, vol. 118, pp. 392–416, 2021.

[7] A. Lohachab, S. Garg, and M. Bilal Amin, "Oclient: On-board client clustering to bridge the interaction gap across multi-fabric adaptive ecosystem," 2021.

[8] E. Buchman, "Tendermint: Byzantine fault tolerance in the age of blockchains," Ph.D. dissertation, University of Guelph, 2016.

[9] E. Abebe, D. Behl, C. Govindarajan, Y. Hu, D. Karunamoorthy, P. Novotny, V. Pandit, V. Ramakrishna, and C. Vecchiola, "Enabling enterprise blockchain interoperability with trusted data transfer (industry track)," in *Proceedings of the 20th International Middleware Conference Industrial Track*, 2019, pp. 29–35.

[10] C. Cachin, M. V. Sorniotti, and T. Weigold, "Blockchain, cryptography, and consensus," *IBM Res., Zürich, Switzerland, Tech. Rep*, vol. 2016, 2016.

[11] C. Gorenflo, S. Lee, L. Golab, and S. Keshav, "Fastfabric: Scaling hyperledger fabric to 20 000 transactions per second," *International Journal of Network Management*, vol. 30, no. 5, p. e2099, 2020.

[12] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer *et al.*, "On scaling decentralized blockchains," in *International conference on financial cryptography and data security*. Springer, 2016, pp. 106–125.

[13] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich *et al.*, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the thirteenth EuroSys conference*, 2018, pp. 1–15.

[14] K. Yamashita, Y. Nomura, E. Zhou, B. Pi, and S. Jun, "Potential risks of hyperledger fabric smart contracts," in *2019 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*. IEEE, 2019, pp. 1–10.

[15] H. Hasanova, U.-j. Baek, M.-g. Shin, K. Cho, and M.-S. Kim, "A survey on blockchain cybersecurity vulnerabilities and possible countermeasures," *International Journal of Network Management*, vol. 29, no. 2, p. e2060, 2019.

[16] Y. Huang, Y. Bian, R. Li, J. L. Zhao, and P. Shi, "Smart contract security: A software lifecycle perspective," *IEEE Access*, vol. 7, pp. 150 184–150 202, 2019.

[17] A. Davenport, S. Shetty, and X. Liang, "Attack surface analysis of permissioned blockchain platforms for smart cities," in *2018 IEEE International Smart Cities Conference (ISC2)*. IEEE, 2018, pp. 1–6.

[18] E. Bellini, Y. Iraqi, and E. Damiani, "Blockchain-based distributed trust and reputation management systems: A survey," *IEEE Access*, vol. 8, pp. 21 127–21 151, 2020.

[19] S. Brotsis, N. Kolokotronis, K. Limniotis, G. Bendiab, and S. Shiaeles, "On the security and privacy of hyperledger fabric: Challenges and open issues," in *2020 IEEE World Congress on Services (SERVICES)*. IEEE, 2020, pp. 197–204.

[20] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *2016 IEEE symposium on security and privacy (SP)*. IEEE, 2016, pp. 839–858.

[21] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi, "Town crier: An authenticated data feed for smart contracts," in *Proceedings of the 2016 aCM sIGSAC conference on computer and communications security*, 2016, pp. 270–282.

[22] B. Beckert, M. Herda, M. Kirsten, and J. Schiffl, "Formal specification and verification of hyperledger fabric chaincode," in *Proc. Int. Conf. Formal Eng. Methods*, 2018, pp. 44–48.

[23] W. Ahrendt, B. Beckert, R. Bubel, R. Hähnle, P. H. Schmitt, and M. Ulbrich, "Deductive software verification–the key book," *Lecture Notes in Computer Science*, vol. 10001, 2016.

[24] S. Kalra, S. Goel, M. Dhawan, and S. Sharma, "Zeus: Analyzing safety of smart contracts." in *Ndss*, 2018, pp. 1–12.

[25] X. Liang, S. Shetty, D. Tosh, P. Foytik, and L. Zhang, "Towards a trusted and privacy preserving membership service in distributed ledger using intel software guard extensions," in *International Conference on Information and Communications Security*. Springer, 2017, pp. 304–310.

[26] C. DeCusatis, M. Zimmermann, and A. Sager, "Identity-based network security for commercial blockchain services," in *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2018, pp. 474–477.

[27] N. Andola, M. Gogoi, S. Venkatesan, S. Verma *et al.*, "Vulnerabilities on hyperledger fabric," *Pervasive and Mobile Computing*, vol. 59, p. 101050, 2019.

[28] S. Mazumdar and S. Ruj, "Design of anonymous endorsement system in hyperledger fabric," *IEEE Transactions on Emerging Topics in Computing*, 2019.

[29] P. Thakkar, S. Nathan, and B. Viswanathan, "Performance benchmarking and optimizing hyperledger fabric blockchain platform," in *2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE, 2018, pp. 264–276.

[30] H. Javaid, C. Hu, and G. Brebner, "Optimizing validation phase of hyperledger fabric," in *2019 IEEE 27th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE, 2019, pp. 269–275.

[31] T. Inagaki, Y. Ueda, T. Nakaike, and M. Ohara, "Profile-based detection of layered bottlenecks," in *Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering*, 2019, pp. 197–208.

[32] H. Gupta, S. Hans, K. Aggarwal, S. Mehta, B. Chatterjee, and P. Jayachandran, "Efficiently processing temporal queries on hyperledger fabric," in *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 2018, pp. 1489–1494.

[33] L. Xu, W. Chen, Z. Li, J. Xu, A. Liu, and L. Zhao, "Locking mechanism for concurrency conflicts on hyperledger fabric," in *International Conference on Web Information Systems Engineering*. Springer, 2020, pp. 32–47.

[34] T. Nakaike, Q. Zhang, Y. Ueda, T. Inagaki, and M. Ohara, "Hyperledger fabric performance characterization and optimization using goleveldb benchmark," in *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 2020, pp. 1–9.

[35] E. Rohrer and F. Tschorsch, "Kadcast: A structured approach to broadcast in blockchain networks," in *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, 2019, pp. 199–213.

[36] N. Berendea, H. Mercier, E. Onica, and E. Riviere, "Fair and efficient gossip in hyperledger fabric," in *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2020, pp. 190–200.

[37] R. H. Kim, H. Noh, H. Song, and G. S. Park, "Quick block transport system for scalable hyperledger fabric blockchain over d2d-assisted 5g networks," *IEEE Transactions on Network and Service Management*, 2021.

[38] J. Sousa, A. Bessani, and M. Vukolic, "A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform," in *2018 48th annual IEEE/IFIP international conference on dependable systems and networks (DSN)*. IEEE, 2018, pp. 51–58.

[39] S. Duan, M. K. Reiter, and H. Zhang, "Beat: Asynchronous bft made practical," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 2028–2041.

[40] M. Ben-Or, B. Kelmer, and T. Rabin, "Asynchronous secure computations with optimal resilience," in *Proceedings of the thirteenth annual ACM symposium on Principles of distributed computing*, 1994, pp. 183–192.

[41] C. Cachin, K. Kursawe, F. Petzold, and V. Shoup, "Secure and efficient asynchronous broadcast protocols," in *Annual International Cryptology Conference*. Springer, 2001, pp. 524–541.

[42] C. Cachin and J. A. Poritz, "Secure intrusion-tolerant replication on the internet," in *Proceedings International Conference on Dependable Systems and Networks*. IEEE, 2002, pp. 167–176.

[43] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song, "The honey badger of bft protocols," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 31–42.

[44] M. Castro and B. Liskov, "Practical byzantine fault tolerance and proactive recovery," *ACM Transactions on Computer Systems (TOCS)*, vol. 20, no. 4, pp. 398–461, 2002.

[45] J. Baek and Y. Zheng, "Simple and efficient threshold cryptosystem from the gap diffie-hellman group," in *GLOBECOM'03.*

IEEE Global Telecommunications Conference (IEEE Cat. No. 03CH37489), vol. 3. IEEE, 2003, pp. 1491–1495.

[46] A. Boldyreva, "Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme," in *International Workshop on Public Key Cryptography*. Springer, 2003, pp. 31–46.

[47] T. Crain, V. Gramoli, M. Larrea, and M. Raynal, "Dbft: Efficient leaderless byzantine consensus and its application to blockchains," in *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*. IEEE, 2018, pp. 1–8.

[48] A. Bessani, J. Sousa, and E. E. Alchieri, "State machine replication for the masses with bft-smart," in *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, 2014, pp. 355–362.

[49] C. Dwork, N. Lynch, and L. Stockmeyer, "Consensus in the presence of partial synchrony," *Journal of the ACM (JACM)*, vol. 35, no. 2, pp. 288–323, 1988.

[50] D. Dolev, C. Dwork, and L. Stockmeyer, "On the minimal synchronism needed for distributed consensus," *Journal of the ACM (JACM)*, vol. 34, no. 1, pp. 77–97, 1987.

[51] J.-P. Martin and L. Alvisi, "Fast byzantine consensus," *IEEE Transactions on Dependable and Secure Computing*, vol. 3, no. 3, pp. 202–215, 2006.

[52] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong, "Zyzzyva: Speculative byzantine fault tolerance," *ACM Transactions on Computer Systems (TOCS)*, vol. 27, no. 4, pp. 1–39, 2010.

[53] P.-L. Aublin, R. Guerraoui, N. Knežević, V. Quéma, and M. Vukolić, "The next 700 bft protocols," *ACM Transactions on Computer Systems (TOCS)*, vol. 32, no. 4, pp. 1–45, 2015.

[54] L. Baird, "Hashgraph consensus: fair, fast, byzantine fault tolerance," *Swirlds Tech Report, Tech. Rep.*, 2016.

[55] F. Xiang, W. Huaimin, S. Peichang, O. Xue, and Z. Xunhui, "Jointgraph: A dag-based efficient consensus algorithm for consortium blockchains," *Software: Practice and Experience*, 2019.

[56] A. E. Gencer, S. Basu, I. Eyal, R. Van Renesse, and E. G. Sirer, "Decentralization in bitcoin and ethereum networks," in *International Conference on Financial Cryptography and Data Security*. Springer, 2018, pp. 439–457.

[57] G. G. Gueta, I. Abraham, S. Grossman, D. Malkhi, B. Pinkas, M. Reiter, D.-A. Seredinschi, O. Tamir, and A. Tomescu, "Sbft: a scalable and decentralized trust infrastructure," in *2019 49th Annual IEEE/IFIP international conference on dependable systems and networks (DSN)*. IEEE, 2019, pp. 568–580.

[58] M. Castro, B. Liskov *et al.*, "Practical byzantine fault tolerance," in *OSDI*, vol. 99, no. 1999, 1999, pp. 173–186.

[59] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted execution environment: what it is, and what it is not," in *2015 IEEE Trustcom/BigDataSE/ISPA*, vol. 1. IEEE, 2015, pp. 57–64.

[60] A. Dhar, I. Puddu, K. Kostiainen, and S. Capkun, "Proximitee: Hardened sgx attestation by proximity verification," in *Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy*, 2020, pp. 5–16.

[61] I. De Oliveira Nunes, X. Ding, and G. Tsudik, "On the root of trust identification problem," in *Proceedings of the 20th International Conference on Information Processing in Sensor Networks (co-located with CPS-IoT Week 2021)*, 2021, pp. 315–327.

Sidharth Quamara is pursuing his Ph.D. at the Department of Computer Engineering, National Institute of Technology Kurukshetra (India). His main area of research is Distributed Ledger Technology, specifically Cryptocurrencies. He received his Master of Technology degree from the Department of Computer Science and Applications, Kurukshetra University (India), in 2011, where his area of research was Aspectized Web Services. His research interests broadly include – Distributed computing, Communication networks, Network and computer security, and Transaction processing.

Awadhesh Kumar Singh received his Bachelor of Technology degree in Computer Science from Madan Mohan Malaviya University of Technology, Gorakhpur, India, in 1988, and his Master of Technology and Ph.D. degrees in Computer Science from Jadavpur University, Kolkata, India, in 1998 and 2004, respectively. He joined the Department of Computer Engineering at the National Institute of Technology, Kurukshetra, India, in 1991, where he is presently a Professor. Earlier, he also served as the Chairman of the department during 2007–2009 and 2013-2015. His research excellence is acknowledged by the paper in journals, IEEE Signal Processing Letters, Journal of Computer Science and Technology, Wireless Personal Communication, Journal of Super Computing, Journal of Information Science and Engineering, Journal of Communication Networks and Information Security, Journal of Applied Evolutionary Computation, and alike. His research interests include distributed algorithms, mobile computing, fault tolerance, and wireless communication.