

Blockchain in Java Refers to Build Scalable, Secure and Interoperable Solutions for Corporate and Government IT Systems

Paresh R. Rathod^{1,*}, Parthiv J. Andharia²

Abstract

In corporate and government IT environments, Blockchain and Java are the modern technology stack that complement each other for building. The Blockchain is the one that presents a distributed, non-repudiable ledger that has the power to bring about transparency, auditability, and trust among people and organizations through the use of Java which is a portable runtime, mature security libraries, and strong integration capabilities with existing systems. When combined, they make it possible to design permissioned blockchain networks, secure smart-contract logic, and deploy scalable middleware on heterogeneous infrastructures from on-premises data centers to cloud-based environments. It mainly talks about using Java for blockchain-centric architecture development that meets the typical enterprise and e-government requirements such as the confidentiality of sensitive data, fine-grained access control, interoperability with legacy systems, and high transaction throughput. It also includes an analysis of the major design decisions like the use of Java cryptography and security APIs, patterns for integrating Java microservices with blockchain nodes, and hybrid on-chain/off-chain data management strategies. Besides that, the paper also proposes a reference architecture where Java-based services provide secure APIs to communicate with a permissioned blockchain layer that serves as a common trust and audit fabric for multiple departments or organizations. By studying this architecture and corresponding use scenarios, the research intends to demonstrate that the deployment of Java-based blockchain solutions can guide the transition of corporate and public-sector IT systems to more secure, transparent, and interoperable digital ecosystems.

Keyword: Blockchain, java, cybersecurity, interoperability, corporate IT, e-government, distributed ledger.

INTRODUCTION

Corporate and government IT systems are progressively being forced to handle sensitive information, control intricate activities, and enable processes that involve more than one stakeholder. Such scenarios call for powerful solutions that assure trust, transparency, and security, even when crossing the borders

*Author for Correspondence

Paresh R. Rathod

E-mail: director@swamigurukul.com

¹Principal, Shree Swaminarayan College of Computer Science, Bhavnagar, Gujarat, India.

²Assistant Professor, Shree Swaminarayan College of Computer Science, Bhavnagar, Gujarat, India.

Received Date: December 15, 2025

Accepted Date: January 16, 2026

Published Date: February 12, 2026

Citation: Paresh R. Rathod, Parthiv J. Andharia. ParkSmart: Blockchain in Java Refers to Build Scalable, Secure and Interoperable Solutions for Corporate and Government IT Systems. Journal of Computer Technology & Applications. 2026; 17(2): 9–17p.

of different organizations. In the past few years, blockchain technology has been considered a potential method for meeting these requirements [1]. The main features of blockchain, such as providing records of transactions that cannot be altered, distributed agreements, and traceable histories of transactions, allow organizations to build systems that are not only capable of detecting fraud but also resistant to unauthorized alterations and operational failures. Furthermore, Java, which is the most preferred programming language in enterprise settings, brings along the advantages of portability, ease of maintenance, and also a wide array of libraries for cryptography, networking, and

integration [2]. Therefore, the cooperation of Java and blockchain will allow organizations to build platforms that are not only scalable but also interoperable to connect their legacy applications, modern microservices, and external partners operating within secure digital ecosystems. Blockchain networks are primarily classified into public, private, and consortium networks. Public blockchains are accessible to everyone and are commonly used for creating digital currencies. Private blockchains limit access only to those who are authorized and are mainly used in the business world [3]. Consortium blockchains are managed by a few organizations and are used for cooperative work among them. Organizations and governments tend to opt for permissioned (private or consortium) models mainly because they provide more control over access, privacy, and adherence to regulations. Java has become the most sought-after programming language for enterprise systems because of its various advantages. First, the Java Virtual Machine (JVM) is one of the main reasons why Java is highly preferred, as it gives developers the flexibility to run their applications on any platform that supports the JVM. Second, Java is also highly considered because of its strong tools and vibrant ecosystem comprising Spring and Jakarta EE, which are the most popular frameworks that ease the development of complex and distributed applications [4].

SECURITY AND TRUST WITH JAVA BASED BLOCKCHAIN

Cryptography and Identity management

Java's cryptography and identity-management skills are the main components of security on Java-based blockchain platforms used in enterprises and e-government. The standard pluggable framework provided by the Java Cryptography Architecture (JCA) and Java Cryptography Extension (JCE) allows the implementation of basic cryptographic techniques—encryption, hashing, and digital signatures—using different providers (e.g., SunJCE, Bouncy Castle, or HSM vendors) [5].

The change in the provider occurs transparently without changing the application code. This provider model is very beneficial for blockchain systems because it allows companies to use software-based providers during the development phase and slowly transition to FIPS-compliant or high-assurance providers during the production phase while maintaining the same Java API and increasing the security assurance level. Key management is a crucial aspect of identity in permissioned blockchains, and Java offers various ways to handle it securely [6]. Java KeyStore (JKS, PKCS#12, or vendor-specific types) enables applications to keep their private keys, public keys, and X.509 certificates inside encrypted containers, which are protected by passwords or integrated with OS-level or hardware-backed keystores. With JCA and related APIs, the administrators can create key pairs, import existing keys, rotate them regularly, and link them to organizational or user identities represented by certificates issued from an internal or external PKI. These certificates can indicate roles, organizational units, or assurance levels, which can then be interpreted by higher-level identity and access management logic before allowing permissions on the blockchain network. Enterprise-grade deployments often require even stronger guarantees, such as resistance to key extraction by malware or insider threats, where hardware security modules (HSMs) or smart cards are used [7]. Java integrates with such hardware through standard interfaces like PKCS#11, allowing cryptographic operations (key generation, signing, decryption) to be performed inside the certified hardware while the Java application only sees opaque key handles. This means that the private keys for validators, ordering services, or high-value signers never leave secure hardware but can still be invoked through familiar JCA and JCE APIs. In a blockchain context, validator nodes or identity gateways implemented in Java can thus offer high-assurance signing of blocks and identity tokens, backed by tamper-resistant hardware anchored in a corporate or government trust hierarchy [8].

Access Control and Network Permission

Blockchain solutions based on Java are especially apt for installations in enterprises and governments, where there is a need for detailed access control and compliance with regulations. In such places, permissioned (private or consortium) blockchain networks are preferred because they allow only authorized users, services, or devices to participate, thereby securing privacy and compliance with regulations [9]. The use of Java makes it possible to implement very strong access control methods, such as role-based access control (RBAC), token-based authentication, and policy enforcement, which are very important for regulating access to blockchain APIs and smart contracts.

Java-based systems utilize RBAC by linking user roles with defined permissions, thus permitting only authorized participants to perform particular actions on the blockchain network. Token-based authentication, with the aid of common standards such as OAuth2 and OpenID Connect, provides safe and scalable authentication for both users and services, thereby ensuring that only the rightful users can engage with the blockchain [10]. These frameworks can be combined in Java applications to facilitate the installation and secure management of identities, preventing non-authorized identities from executing transactions or accessing confidential data.

Policy enforcement escalates through the adoption of policy enforcement points (PEPs) and policy decision points (PDPs), which directly automate access decisions based on pre-existing security policies. The main function of PEPs is to intercept access requests and then send them forward to PDPs, where they are evaluated against the stored policies, and a decision (either allow or deny) is issued back to the requester. By this method, it is guaranteed that all transactions and data exchanges are subjected to rigid security policies, which in turn, augment, trust, auditability, and operational efficiency in the case of sensitive environments [11]. The versatility of the Java development environment makes it possible to place these components inside blockchain gateways, APIs, or middleware, thus delivering a total access control layer for both corporate and public-sector IT systems.

SCALABILITY AND PERFORMANCE IN ENTERPRISE & GOVERNMENT CONTEXT

Java Microservices and Blockchain Integration

Java microservices represent a robust architectural pattern for establishing integration with the blockchain nodes, thereby fostering the development of solutions that are efficient, scalable, and maintainable in the contexts of both enterprises and governments. By using REST, gRPC, or message queues, Java microservices can communicate with blockchain networks without the need to incorporate blockchain logic into business applications, thereby keeping concerns apart and facilitating modular design [12]. Within this integration, microservices serve as intermediaries between user-facing applications and blockchain nodes.

To give an example, a microservice might provide RESTful APIs for the purpose of getting transaction data from client applications and then sending it to a blockchain node through standards like HTTP/JSON or gRPC. User authentication and business logic would still be present in the microservices, but blockchain-specific operations, such as transaction signing, block submitting, and event listening, would be left to integration services or adapters [13]. This separation guarantees that each service can grow or scale independently, and the entire system can manage high request volumes without experiencing performance drops. Message queues, including RabbitMQ and Apache Kafka, are very effective for managing busy scenarios, such as corporate portals or government applications for citizens. Message queues use an asynchronous process that decouples blockchain operations from user-facing services, allowing the system to queue and process transactions in batches. This leads to an improvement in responsiveness and the provision of resilience over network issues or blockchain node downtime [14].

For instance, when a blockchain node becomes unavailable for a short period, the transactions can be queued and processed when the node is back online, thus ensuring no data loss and service continuity. This pattern remains significant for throughput and responsiveness to be retained in high-volume environments. Java-based systems, by the deployment of microservices and asynchronous communication, can serve large user bases very well, thus guaranteeing fast response times and dependable service delivery. Furthermore, this strategy makes updating and maintenance more manageable because the blockchain integration logic can be modified or substituted without disrupting the main business logic. Hence, Java microservices are the perfect choice for scalable blockchain solutions in enterprise and government contexts [15].

Optimizing Throughput and Latency

In both enterprise and government areas, blockchain systems face serious performance difficulties, such as consensus overhead, block propagation delay, and transaction validation costs. These problems might cause latency to increase and throughput to decrease, which is not good at all in high-volume places such as corporate portals and citizen-facing government apps. Just to mention an example, consensus mechanisms, that is, Proof-of-Work or Proof-of-Stake, require a while for nodes to agree on the validity of transactions, thus, the whole network can be slowed down [16].

Besides the above-mentioned time, there is block propagation delay or the time it takes for a new block to be acknowledged throughout the network—this also affects the system's speed, especially in widespread installations. In addition, there are cryptographic checks and smart contract execution, which means transaction validation, where the performance gets degraded due to the added computational overhead. Java-based solutions present a variety of approaches to address these issues by optimizing throughput and latency. One of the main tactics used in this area is caching, which stores the blockchain data that is frequently accessed (such as identity information or transaction status) in fast in-memory caches like Redis or Ehcache. This practice lessens the need for on-chain queries more than once; therefore, it lowers the latency and makes the system more responsive. Another efficient way to deal with the problem is the batching of transactions, which allows the grouping of a number of operations together and submitting them to the blockchain in a single request.

This not only helps reduce interactions with the blockchain but also minimizes network congestion and spreads the cost of consensus and validation over several transactions, thus leading to an overall improvement in throughput. Asynchronous processing is a technique that has been applied to Java-based blockchain systems. When blockchain operations are transferred to background processes or message queues, user-facing services can still respond to requests quickly, even if the blockchain is slow to process. Gracing user experience with no regard to blockchain delay, this method guarantees that the system maintains its responsiveness even when it is overloaded or during times of network congestion. The multithreading and asynchronous programming support in Java (such as through *CompletableFuture* or reactive frameworks) makes it a candidate for such implementations. The combination of optimization methods—caching, batching, and asynchronous processing—allows Java-based blockchain solutions to maintain high throughput and low latency even in the most challenging enterprise and government settings. These methods, by lessening the effect of consensus overhead, block propagation delay, and transaction validation costs, ensure that blockchain applications are scalable, responsive, and efficient for large user bases.

INTEROPERABILITY FOR CORPORATE & GOVERNMENT IT

Integrate Legacy System

Legacy systems are still a necessity along with Java-based platforms and non-Java technologies in several corporate and government IT setups, and they are behind the curtain from the start to the end of operations. These legacy systems are generally monolithic, run-on aging programming languages, and use inflexible databases, which are obstacles to their modernization and new technology integration. Nevertheless, the demand for secure, transparent, and efficient data exchange has prompted organizations to consider blockchain as a shared trust layer that can unify these isolated systems, allowing smooth interoperability without the need for extensive code rewrites. Blockchain has been developed as a decentralized, unchangeable ledger that guarantees data integrity and auditability for new applications and those that have already been built. Connecting blockchain with existing systems, companies will be able to share data with newer systems in a manner that is secure, unalterable and authenticated thus eliminating the chances of fraud and unauthorized access.

This method is greatly beneficial to enterprises and governments, where compliance with regulations, data protection, and uninterrupted operation are of utmost importance. Java adapters, connectors, and integration frameworks, such as Enterprise Service Buses (ESBs) and API gateways, play a key role in

keeping data and events in sync between old applications and blockchain networks. These middleware solutions serve as a channel that maintains the data format of the old system and the communication protocol for the blockchain, meaning they can translate and convert the legacy system data and its communication protocol into blockchain-compatible formats and vice versa. For instance, an ESB can direct transaction data from an outdated ERP system to a blockchain node, whereas an API gateway can make available blockchain services to legacy applications through standard REST or SOAP interfaces.

In this way, organizations can take advantage of the immutable and decentralized nature of blockchains while simultaneously relying on the stability and functionality of their existing systems. In addition, these integration patterns provide for gradual rollouts, which, in turn, lower the risks and alterations that come with a complete system upgrade. Companies may introduce blockchain features gradually, beginning with the most important processes and then expanding, if necessary, thus guaranteeing a seamless transition with little or no downtime. With the help of Java-based adapters and connectors, companies can keep their legacy infrastructure under control while simultaneously benefiting from the characteristics that blockchain provides in terms of data sharing, verification, and recordkeeping. This method creates an environment where trust is built, security is improved, and cooperation across both old and new systems is made efficient; hence, it is a perfect solution for corporate and government IT settings.

Cross Organization Data Sharing

A consortium blockchain allows several organizations (e.g., government departments, agencies, or companies) to hold consistent, non-tampered records and, at the same time, have local control over their respective internal systems. In contrast to public blockchains, consortium blockchains are controlled by a group of reliable participants, who, in turn, keep full control over their own hardware and software as well as data processing practices. This setup is perfect for situations when organizations want to work together but at the same time require to keep things like privacy, regulatory compliance, and independence of operations from each other. The members of a consortium blockchain can keep a common record of their transactions, which means that the organizations will always have access to the same trustworthy documents, thus promoting transparency and confidence in their relations. Java applications perform a vital function in allowing seamless and trustworthy exchange of data among the different organizations participating in the consortium blockchains.

They may impose common data formats, validation criteria, and interoperability standards—like JSON or XML schemas—so that the data exchanged among the organizations is uniform, precise, and fit for the purpose of the regulatory authorities. For example, middleware based on Java can first verify the data coming in against the schemas that have been set out before they are recorded on the blockchain, thereby eliminating any chances of errors and ensuring that all the participants are following the same data standards. This method, in turn, eliminates confusion, reduces conflicts, and improves efficiency in cooperation, as organizations will be confident that the data they share are both legitimate and protected. Moreover, Java applications can utilize complex access control mechanisms, such as role-based permissions and attribute-based access control, to ensure that only the designated users and systems are allowed to either view or change specific data stored in the consortium blockchain.

This way, companies are able to keep tight control over their confidential information and at the same time enjoy the transparency and auditability that the shared ledger has offered. Through the adoption of Java's strong security libraries and integration frameworks, consortium blockchain solutions can be capable of supporting intricate, multi-organizational workflows, which in turn secures the data exchange while making it efficient. In brief, consortium blockchains with the assistance of Java-powered apps are a robust platform for sharing data across organizations in the case of enterprises and governments. These solutions allow organizations to work together efficiently and at the same time

control their data locally, comply with regulations, and ensure security by enforcing data standards, validation rules, and access controls.

CHALLENGES & BEST PRACTICES.

Technical and Organizational Challenges

Merging blockchain technology into the IT infrastructure of corporate and government entities poses numerous technical and organizational difficulties, among which the integration of legacy systems is deemed the most crucial. Quite a few companies still depend on old systems, which are mainly constructed on outdated technologies like COBOL, inflexible databases, and limited APIs, and thus not capable of interfacing with modern blockchain architectures. These systems are often designed in a single-layer architecture, so even small modifications can cause major disruptions, and their unchanging schemas make it hard to map and migrate data to blockchain networks.

When legacy systems fail to interoperate with blockchain platforms, the result is a compatibility issue, data being locked in silos, and poor workflow, which all require organizations to incur the expense of custom middleware, API gateways, or enterprise service buses (ESBs) to connect the two. Another significant challenge is the skills gap, as blockchain and Java-based blockchain development require highly specialized skills that are probably not available among the existing IT teams. This skill shortage can prolong the implementation process and increase reliance on outside professionals, thus making the entire project costlier and time-consuming. Besides, governance of shared ledgers is already a major problem, as several companies have to reach an agreement on the roles, responsibilities, and rules for the nodes and data that participate and are accessed. In the absence of proper governance frameworks, arguments may arise about who owns the data, who has the right to access it, and who has to comply with laws, thus creating a situation of distrust and non-cooperation in the data-sharing process.

Evolving regulatory requirements are a major factor that complicates the integration of blockchain technology, as companies are forced to adapt to new compliance mandates while not impeding their operational stability. It is possible that the existing infrastructure will not be able to comply to the new requirements, which will put the organizations at risk of being caught in the regulators' net and incurring fines. To overcome this, the organizations have to first clarify the duties and tasks of all the stakeholders and then put in place a strong governance policy, and lastly carry out a comprehensive training program. The phased approach to integration, expert knowledge, and secure and interoperable solutions are the three main factors that will enable organizations to integrate blockchain technology while simultaneously keeping their legacy systems stable and compliant.

Security Risks and Mitigation

Blockchain technology does not automatically eliminate all security risks, and Java-based blockchain applications can still be exposed to common threats, such as injection risks, misconfiguration, and insecure key storage. Although the absolute trustworthiness and decentralization of blockchain have been its main strengths, the Java code that communicates with blockchain nodes and handles business logic still has to deal with vulnerabilities such as cross-site scripting (XSS), insecure deserialization, and incorrect access controls.

One such scenario is where Java apps employ weak or outdated encryption methods or do not validate input properly; in such cases, attackers will find it easier to manipulate the situation and thus be able to either compromise data integrity or gain unauthorized access. To reduce the aforementioned risks, organizations must implement secure coding practices that specifically address Java and blockchain technologies. Replacing older encryption methods, such as DES, with current ones, such as AES, which are strong and up-to-date encryption algorithms, is one of the measures to be taken, along with input data validation, to impede injection attacks. Furthermore, developers can completely prevent exposure caused by insecure deserialization if they primarily rely on libraries and frameworks regarded as secure, such as Jackson or GSON, for their serialization/deserialization work. In addition, robust access control

methods, such as role-based access control (RBAC), should be developed and enforced so that only individuals or services with the right credentials can access sensitive blockchain APIs and smart contracts.

Regular code reviews and penetration testing are necessary steps in the process of detecting and eliminating vulnerabilities before they can be exploited. The use of automated security scanning tools can identify the most common coding errors and insecure setups, while manual code reviews carried out by security experts can highlight more delicate problems. Penetration testing, an imitation of real-world attacks, enables organizations to determine the strength of their Java-based blockchain applications and the improvements to be made. Moreover, the proper management of keys is of great significance in securing Java-based blockchain applications.

First, to prevent unauthorized access, sensitive cryptographic keys should be stored in an extremely secure manner, with the use of hardware security modules (HSMs) or secure key storage solutions being the most recommended options. Moreover, developers are expected to not embed any credentials within the source code; rather, they should apply secure configuration management practices. Overall, by adopting these practices, organizations will not only significantly lessen the chance of a security breach but also keep their Java-based blockchain applications very robust and immune to the changing threat landscape.

Recommended Design Practices

It is suggested that the architectural design of Java-based blockchain applications should be done in such a way as to have a layered architecture along with the use of standardized APIs and monitoring systems to be very reliable, which would guarantee the three aspects of scalability, interoperability, and security. A layered architecture permits the splitting of the presentation, business logic, and blockchain interaction modules so that the development, testing, and scaling of each layer can occur independently. For instance, the presentation layer takes care of the user interfaces, the business logic layer takes application-specific rules as input and provides the output as application-logic-processed data, and the blockchain interaction layer governs the communication with blockchain nodes through REST, gRPC, or message queues.

This division of concerns leads to better maintainability, easier upgrades, and new technologies being majorly plugged into the system without interrupting the main business functions. Interoperability is highly dependent on standardized APIs and protocols, which allow different systems and services to communicate without interruption. In this way, Java blockchain applications that use widely accepted standards such as REST, JSON, or XML will be able to communicate with both old and new systems consisting of microservices, thus the data trading will be backed up as uniform and trustworthy. Furthermore, standardization provides easier access to third-party services and ensures compliance with regulations, thus making inter-company data sharing less complicated.

The abnormal behavior of a blockchain network can be detected and responded to promptly using monitoring, logging, and alerting methods. Java services are supposed to provide complete logging for the purpose of following transactions, errors, and system events, and have monitoring tools to gather metrics like transaction throughput, latency, and node health. Automated alerting can inform admins about possible issues, such as failed transactions or unusual activities, thus enabling rapid intervention and lessening downtime. Such practices make the system more reliable, maintain the system proactively, and ensure that Java-based blockchain applications are robust and responsive to fluctuating enterprise and government settings.

CONCLUSION

The combination of Java and blockchain technology creates a strong foundation for modern IT systems in the corporate and government sectors, offering digital ecosystems that are secure, scalable, and interoperable. The combination of Java's powerful security libraries, adaptable architecture, and

great integration with blockchain's openness, auditability, and trustworthiness provides a superior option for permissioned networks that require confidentiality, meticulous access control, and smooth integration with legacy systems. By using Java's encryption and identity management features, companies can achieve strong authentication and non-repudiation, which are vital for enterprise and government applications.

Access control methods, such as role-based access and token-based authentication, not only protect privacy but also ensure compliance with regulations. In addition, microservices and asynchronous processing contribute to the overall scalability and performance of the system. Interoperability is made possible through Java adapters and integration frameworks that facilitate the interaction of legacy systems with blockchain networks without significant rewrites. Consortium blockchains allow a number of companies to have shared records that are consistent, but the control is still local, and Java applications dictate data standards and validation rules for safe and reliable data exchange among different organizations.

Organizations will still have to address technical and organizational issues. These include the integration of legacy systems, shortages of skilled personnel, governance, and the constantly changing demands of regulators. Security attacks, such as injections and bad key storage, are some of the risks that can be controlled by using secure coding, conducting regular reviews, penetration testing, and managing keys strictly. Recommended design practices, such as layered architecture, standardized APIs, and comprehensive monitoring, provide both robustness and maintainability. In short, Java-based blockchain technologies are a mighty platform for creating safe, scalable, and interoperable digital ecosystems that will lead to the next wave of digital transformation in business as well as government.

REFERENCES

1. Singh AK, Kumar VG, Hu J, Irfan M. Investigation of barriers and mitigation strategies to blockchain technology implementation in construction industry: an interpretive structural modeling approach. *Environmental Science and Pollution Research*. 2023 Aug;30(38):89889-909.
2. Aro OE, Nweze M, Avickson EK, Aro OE, Nweze M, Avickson EK. Blockchain technology as a tool for corporate governance and transparency. *International Journal of Science and Research Archive*. 2024;13(01):2479-93.
3. Gupta, M. (2017) *Blockchain for Dummies*. Wiley. - References - Scientific Research Publishing. Scirp.org. 2017. Available from: <https://www.scirp.org/reference/referencespapers?referenceid=3813555>.
4. Blockchain, Fintech, IT. *Economy.zg.ch*. 2023. Available from: https://economy.zg.ch/en/cluster/blockchain_fintech_it
5. Salama R, Al-Turjman F. Distributed mobile cloud computing services and blockchain technology. In *Computational Intelligence and Blockchain in Complex Systems 2024 Jan 1* (pp. 205-214). Morgan Kaufmann.
6. Owolabi BO, Owolabi FA. Blockchain-powered health innovation information systems for secure, interoperable, and privacy-preserving healthcare data management. *Int J Sci Res Arch*. 2025;15(2):1680-98.
7. AlSalamah S. VCAC: A Blockchain-Based Virtual Care Access Control Model for Transforming Legacy Healthcare Information Systems and EMRs into Secure, Interoperable Patient-Centered Virtual Hospital Systems. *Information*. 2025 Nov 11;16(11):972.
8. Chevallereau B, Carter G, Sneha S. Voice biometrics and blockchain: Secure interoperable data exchange for healthcare. *Blockchain in Healthcare Today*. 2019;2.
9. Lokre A, Faratkhan S, Patil S, Bagwan-Sheikh S. Valtora - Secure File Management System Using Blockchain, IPFS, and Smart Contracts. *International Journal For Multidisciplinary Research*. 2025 Nov 11;7(6). Available from: <https://www.ijfmr.com/research-paper.php?id=59624>
10. Patwe S, Mane SB. Blockchain-enabled secure and interoperable authentication scheme for metaverse environments. *Future Internet*. 2024 May 11;16(5):166.

11. Zhao Y, Song Z, Song J, Tong H, Xing Z, Jiang R. A privacy-preserving and secure editable blockchain model for government data sharing: Y. Zhao et al. *The Journal of Supercomputing*. 2025 May 27;81(8):898.
12. Nazir A, He J, Zhu N, Anwar MS, Pathan MS. Enhancing IoT security: a collaborative framework integrating federated learning, dense neural networks, and blockchain. *Cluster computing*. 2024 Sep;27(6):8367-92.
13. Guayasamín A, Fuertes W, Carrera N, Tello-Oquendo L, Suango V. Blockchain-empowered e-ticket distribution system for secure and efficient transactions, validation, and audits. *Annals of Telecommunications*. 2025 Nov;80(11):1195-220.
14. Ahmed S. Enhancing data security and transparency: The role of blockchain in decentralized systems. *International Journal of Advanced Engineering, Management and Science*. 2025;11(1):593258.
15. Jaiswal IA. Integrating AI into enterprise Java applications for secure high performance and scalable systems. *International Journal of Computational and Experimental Science and Engineering*, 11 (4). <https://doi.org/10.22399/ijcesen>. 2025;4086.
16. Yahyaoui A, Kotel S, Hamel L, Lahouij A, Maraoui Kamoun R. Secure and scalable IoT healthcare with a permissioned blockchain and encryption techniques: A. Yahyaoui et al. *Computing*. 2025 Aug;107(8):172.