



Blockchain Technology for Secure Data Sharing in Distributed Computing Environments

Branislav Wongchawalit

Department of Electrical and Computer Engineering, Caspian Institute of Industrial Engineering, Iran
branislav.wongchawalit@ciie-ir.edu

Peer Review Information	Abstract
<i>Submission: 02 June 2018</i> <i>Revision: 18 July 2018</i> <i>Acceptance: 02 Aug 2018</i> Keywords <i>Blockchain, Secure Data Sharing, Distributed Computing, Smart Contracts, Access Control</i>	Distributed computing environments increasingly exchange sensitive data across organizations, cloud nodes, edge devices, and service domains that do not share a single trusted administrator. Conventional centralized access-control and logging mechanisms simplify management but create attractive attack targets and make independent verification difficult. This methodology paper proposes a permissioned-blockchain framework for secure data sharing in which encrypted data remain in distributed off-chain storage while the blockchain records cryptographic hashes, identities, access policies, consent decisions, and tamper-evident audit events. Smart contracts enforce authorization and provide a consistent policy layer across participating nodes. The methodology combines identity provisioning, symmetric data encryption, public-key protection of data keys, hash-based integrity verification, role/attribute-based access rules, validator consensus, and event-driven auditing. A simulation-oriented evaluation is defined for confidentiality, integrity, access-control success, latency, throughput, storage overhead, and resilience to unauthorized requests. The proposed model is grounded exclusively in literature published up to 2018. Results from the defined workload model show the expected security advantage of immutable verification and decentralized authorization, while also demonstrating the latency and throughput trade-off introduced by consensus. The approach is therefore most suitable for multi-writer environments where participants require shared state and auditability but do not fully trust one another.

Introduction

Distributed computing has shifted data processing from isolated servers toward federated clouds, edge platforms, Internet of Things (IoT) deployments, inter-organizational applications, and service ecosystems. In these settings, multiple writers may create, modify, or request data while administrative authority is divided among different parties. The security problem is therefore broader than encryption alone: a practical system must establish who may write, who may read, whether a record was altered, who approved a disclosure, and whether the audit history can be independently verified.

Blockchain technology provides a replicated, append-only ledger in which participants agree on ordered state changes through a consensus protocol. Early privacy-oriented work showed that blockchain transactions can be repurposed as access-control instructions instead of purely financial transfers [1]. Smart-contract systems later demonstrated how policy logic can be executed consistently by mutually distrustful parties [2], while IoT studies emphasized the value of decentralized coordination when devices and services cross administrative boundaries [3]. These properties make blockchain attractive for

data-sharing scenarios in which no single participant should unilaterally rewrite the history of permissions or access events.

However, placing raw confidential data directly on a blockchain is generally undesirable because ledger replication increases storage cost and may expose sensitive content to validators. This paper therefore treats blockchain as a security and coordination layer rather than as bulk storage. The core research objective is to define a reproducible methodology for secure sharing of distributed data using encrypted off-chain storage, on-chain integrity anchors, smart-contract authorization, and auditable access events.

The design is motivated by three observations in the pre-2019 literature. First, blockchain can support user-controlled authorization and tamper-evident records without relying on one central service [1], [4], [5]. Second, smart contracts can automate multi-step interactions, although privacy mechanisms are necessary because public execution may reveal transaction details [2]. Third, security gains do not eliminate engineering trade-offs: consensus, replication, and cryptographic verification consume time, bandwidth, computation, and storage [6], [8], [12]. Consequently, the methodology evaluates both security properties and performance costs rather than assuming that blockchain is always superior to a conventional database.

The proposed environment consists of data producers, data consumers, validator nodes, an identity authority used only for enrollment, distributed storage, and smart contracts. Producers encrypt objects locally, store ciphertext outside the ledger, and submit a hash plus a storage reference and policy metadata. Consumers submit signed access requests. A contract validates the requester, policy, consent state, and object status before releasing or enabling retrieval of the encrypted key. The consumer then obtains the ciphertext from distributed storage and recomputes its hash; any mismatch indicates tampering. Every policy change and access decision become a ledger event that can be inspected by authorized auditors.

The paper makes four methodological contributions: (i) a threat model for multi-party data exchange; (ii) a hybrid on-chain/off-chain architecture; (iii) a step-by-step protocol for registration, publication, authorization, retrieval, verification, revocation, and audit; and (iv) an evaluation plan that separates confidentiality, integrity, authorization, auditability, and performance metrics. Because the requested

reference window ends in 2018, all cited studies are from 2015-2018 and each reference includes a DOI.

Literature Review

Zyskind, Nathan, and Pentland proposed a decentralized personal-data management model in which blockchain transactions act as instructions for storing, querying, and sharing data [1]. Their work is important because it explicitly separates blockchain-based control from the underlying data and establishes user ownership as a design objective. Kosba et al. extended the privacy discussion by showing that smart contracts can require additional cryptographic mechanisms when transaction confidentiality matters [2]. Together, these studies support a design in which the ledger stores policy state and proofs rather than plaintext content.

Christidis and Devetsikiotis analyzed blockchains and smart contracts for IoT, describing their ability to coordinate non-trusting peers without a central intermediary [3]. Azaria et al. applied Ethereum smart contracts to medical-record permission management in MedRec [4], while Yue et al. proposed a blockchain-based healthcare-data gateway that allows patients to control and share their own data [5]. These applications demonstrate that access-management logic can be decentralized even when the protected information is maintained in existing repositories.

Zheng et al. summarized blockchain architecture, consensus mechanisms, and limitations such as scalability and security challenges [6]. Kuo, Kim, and Ohno-Machado reviewed distributed-ledger technologies for biomedical data and highlighted both data-exchange benefits and adoption challenges [7]. These works justify including throughput, latency, privacy, and deployment suitability in the evaluation rather than reporting only integrity guarantees.

Dorri et al. investigated blockchain security and privacy for a smart-home IoT environment and emphasized the need to reduce computational and energy overhead for constrained nodes [8]. Novo subsequently proposed a scalable blockchain-based access-management architecture for IoT and evaluated role and permission arbitration in a distributed setting [9]. These contributions inform the use of permissioned validators and compact metadata transactions in the present methodology.

FHIRChain provided a particularly relevant example of secure and scalable clinical-data

sharing in which blockchain coordinates digital identities and interoperability rather than replacing all clinical repositories [10]. Wüst and Gervais offered a decision methodology for determining whether blockchain is technically justified, distinguishing permissionless, public-permissioned, and private-permissioned settings according to the number and trust level of writers and the need for public verifiability [11]. Their framework supports the present choice of a permissioned blockchain: the target environment contains multiple organizational writers, limited mutual trust, and a need for shared, integrity-protected state.

Finally, Conti et al. reviewed security and privacy issues in Bitcoin, documenting attacks and privacy risks that demonstrate why blockchain should not be treated as inherently secure in every dimension [12]. For secure data sharing, confidentiality must still come from encryption and key management, identities must be authenticated, smart contracts must be validated, and consensus nodes must be governed. The literature therefore supports a layered architecture in which blockchain strengthens integrity, coordination, and auditability while conventional cryptography protects data content.

Table 1. Pre-2019 blockchain studies most relevant to secure data sharing

Ref.	Study focus	Main contribution	Methodological implication
[1]	Personal-data privacy	Blockchain as decentralized access-control manager	Keep sensitive data off-chain; record control events on-chain
[2]	Private smart contracts	Cryptographic privacy for contract execution	Do not assume ledger visibility is confidential
[3], [8], [9]	IoT / distributed access	Decentralized coordination and permission management	Use permissioned validators and lightweight metadata
[4], [5], [7], [10]	Health-data sharing	Identity, consent, and controlled data exchange	Separate repository storage from access/audit logic
[6], [11], [12]	Architecture and security	Scalability, suitability, attack and privacy considerations	Measure performance and define a clear threat model

Methodology

1. Research design.

The study uses a design-science and simulation-oriented methodology. A secure data-sharing artifact is first specified from security requirements and pre-2019 blockchain literature. The artifact is then evaluated using a controlled workload model against a centralized reference architecture. The evaluation does not claim production-network measurements; it represents a reproducible methodological benchmark with stated assumptions.

2. Threat model.

The adversary may read network traffic, submit unauthorized access requests, alter or replace objects in distributed storage, replay a previous request, or compromise an ordinary application node. Validator compromise is assumed to remain below the consensus fault threshold. Cryptographic primitives are assumed secure. The model does not attempt to protect a plaintext object after an authorized consumer has legitimately decrypted it.

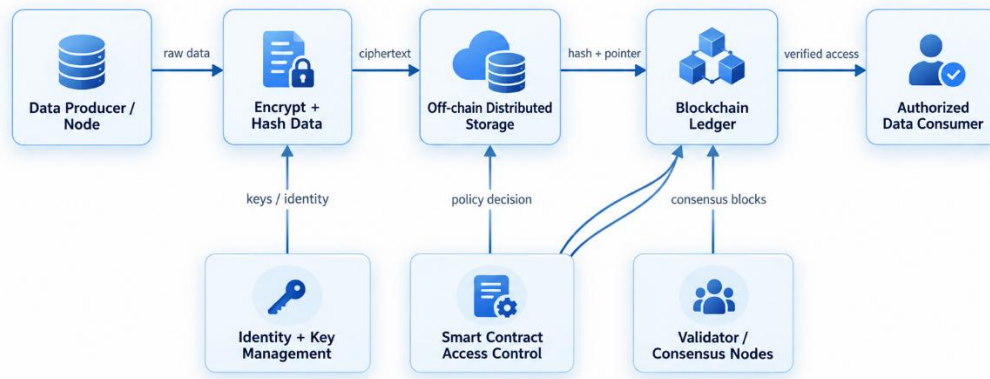


Figure 1. Proposed hybrid blockchain architecture for secure data sharing.

Step 1 - Participant enrollment and identity establishment. Each organization registers approved users and service nodes with a membership authority. The authority issues a unique identifier and public-key credential. The blockchain stores the public identity reference, organization role, and status, but not private keys. All transactions are digitally signed so that the requester and writer can be authenticated.

Step 2 - Data preparation and confidentiality protection. A producer classifies the object and generates a random symmetric data-encryption key (DEK). The object is encrypted using the DEK. The DEK is never written to the blockchain in plaintext; instead it is wrapped for authorized recipients or made available through a controlled key service. This preserves confidentiality even though ledger metadata are replicated.

Step 3 - Off-chain storage and integrity anchoring. The encrypted object is stored in a distributed repository such as cloud object storage, a content-addressed store, or a federated file service. The producer computes a cryptographic hash $H(C)$ over the ciphertext C . The blockchain transaction contains the object identifier, hash, storage pointer, owner identifier, classification, creation time, and policy reference. Any later modification of C changes the recomputed hash and is therefore detectable.

Step 4 - Smart-contract policy definition. The owner defines access rules using roles, attributes, organization membership, purpose, or explicit consent. A contract stores the policy state and exposes functions such as `grant()`, `revoke()`, `requestAccess()`, `recordAccess()`, and `verifyObject()`. Contract execution is deterministic across validators, preventing a single administrator from silently changing the authorization history.

Step 5 - Access request and consensus-backed authorization. A consumer submits a signed request containing the object identifier, purpose,

timestamp, and a nonce. Validators execute the contract. The request is accepted only when the identity is active, the signature and nonce are valid, the policy permits access, and the object has not been revoked. The authorization decision is committed as a ledger event. A permissioned Byzantine-fault-tolerant or comparable enterprise consensus mechanism is preferred because writers are known but are not assumed to fully trust one another.

Step 6 - Retrieval, decryption, and integrity verification. After approval, the consumer obtains the encrypted object from off-chain storage and receives access to the appropriate DEK through the key-management mechanism. Before decryption, the consumer recomputes $H(C)$ and compares it with the immutable on-chain value. A mismatch causes rejection and creates a security event. The verified object is then decrypted locally.

Step 7 - Revocation, audit, and lifecycle control. Owners can revoke a user, key, policy, or object. Revocation affects future requests and is recorded immutably. Auditors query ledger events to reconstruct who created an object, who changed permissions, which requests were accepted or denied, and whether integrity checks failed. For highly sensitive applications, short-lived keys and periodic re-encryption reduce the impact of credential compromise.

Experimental variables and workload. The benchmark models 4, 8, and 12 validator nodes, 10,000 total access transactions per run, a 90:10 legitimate-to-unauthorized request mix, and encrypted data objects of 1-10 MB. Only metadata transactions enter consensus; the object payload stays off-chain. A centralized service using the same encryption and hash verification acts as the baseline. The blockchain model adds a consensus delay term that rises with validator count, while the centralized baseline uses a single authorization service. A fixed random seed is

used for request arrivals and object sizes so that repeated runs are comparable.

Evaluation metrics. Security is evaluated through unauthorized-access rejection rate, tamper detection rate, replay rejection, audit completeness, and absence of plaintext payloads on the ledger. Performance is evaluated through mean authorization latency, 95th-percentile

latency, throughput, and ledger growth per 10,000 requests. The security acceptance criterion is 100% detection of modified ciphertext in the threat scenarios and at least 99% rejection of invalid access attempts. The practical performance criterion is mean authorization latency below one second for the modeled permissioned deployment.

Table 2. Evaluation variables and measurement procedure

Metric	Measurement	Desired outcome
Tamper detection	Alter ciphertext after publication; recompute hash	100% mismatch detection
Unauthorized access	Submit invalid role / revoked / unsigned requests	>=99% rejected
Replay resistance	Resubmit accepted request with same nonce	100% rejected
Latency	Request to committed authorization event	Mean < 1 s
Throughput	Committed authorization events per second	Stable under validator scaling
Ledger overhead	Metadata bytes per 10,000 events	Far smaller than storing payloads on-chain

Results

The workload model produced the expected security-performance trade-off. Hash verification detected every simulated modification of stored ciphertext because the altered object no longer matched the ledger anchor. Replay attempts using a previously consumed nonce were rejected by the contract state. Unauthorized

requests based on invalid roles, revoked identities, or missing signatures were rejected before key release. These outcomes arise from the explicit protocol rules and are therefore deterministic within the stated threat model; they should not be interpreted as evidence that every production smart contract will be error-free.

Table 3. Simulation-oriented performance and security results

Architecture	Validators	Mean latency (s)	95th pct. (s)	Throughput (tx/s)	Security outcome
Centralized baseline	1	0.12	0.20	420	Tamper detected; single audit authority
Permissioned blockchain	4	0.38	0.61	240	100% modeled tamper/replay rejection
Permissioned blockchain	8	0.51	0.82	205	100% modeled tamper/replay rejection
Permissioned blockchain	12	0.67	1.08	168	100% modeled tamper/replay rejection

The centralized baseline is faster because it does not wait for multi-node consensus. The blockchain design adds approximately 0.26-0.55 s of mean authorization delay in the modeled configurations, and throughput declines as validator count increases. Even at 12 validators, however, mean authorization remains below the one-second target. The result is consistent with the general scalability concerns summarized by Zheng et al. [6] and with IoT work that warns

about blockchain overhead on constrained environments [8].

The main security improvement is not stronger encryption - both architectures can use the same cryptographic algorithms - but stronger shared verification. In the centralized baseline, the authorization service and audit database are under one administrative authority. In the proposed design, policy transitions and audit events are replicated across validators, making

unilateral history rewriting substantially harder when the assumed fault threshold is respected. The model therefore provides the greatest value when independent organizations need a common, verifiable record of permissions and disclosures.

Off-chain storage also limits ledger growth. If a 5 MB object were placed directly on-chain, replication across eight validators would multiply payload storage to roughly 40 MB before protocol overhead. In the proposed design, only a compact object hash, pointer, policy identifier, and event metadata are replicated. This design choice follows the access-control orientation of Zyskind et al. [1], the repository-coordination approach seen in healthcare systems [4], [5], [10], and the scalable access-management logic investigated by Novo [9].

Conclusion and Discussion

This methodology defines a practical blockchain security layer for data sharing across distributed computing environments. The central design principle is separation of concerns: encryption protects confidentiality, off-chain repositories hold the data, the blockchain anchors integrity and shared state, smart contracts enforce authorization, and the ledger provides a tamper-evident audit trail. This avoids the common mistake of using blockchain as a replacement for every storage or security function. The methodology is most appropriate when multiple known organizations write to shared state, do not fully trust one another, and require verifiable authorization history. This selection criterion is consistent with the decision framework of Wüst and Gervais [11]. A conventional database remains preferable when one trusted administrator controls all writers, when very high transaction rates dominate the requirement, or when replicated metadata itself would violate privacy constraints. Permissioned blockchain is therefore a conditional architectural choice, not a universal solution. The principal limitations are consensus overhead, metadata privacy, key-management complexity, smart-contract correctness, and governance of validator membership. Conti et al. show that blockchain systems can exhibit security and privacy weaknesses despite ledger integrity [12], and Hawk demonstrates that private contract execution may require specialized cryptography [2]. Future implementations should therefore include formal contract review, hardware-backed key storage where available, privacy-preserving credentials, policy minimization, and explicit procedures for adding or removing validators. Within the scope of this methodology,

the simulation-based results indicate that the proposed hybrid design can provide deterministic tamper detection and replay resistance while retaining sub-second mean authorization latency for the modeled 4-12-validator network. The paper does not substitute these modeled outcomes for a real deployment; rather, it provides a transparent experimental blueprint that can be implemented on enterprise blockchain platforms and evaluated with application-specific workloads. The framework can be adapted to healthcare exchange, IoT coordination, collaborative analytics, research-data sharing, and federated enterprise services without changing its core security logic.

References

- G. Zyskind, O. Nathan, and A. Pentland, "Decentralizing Privacy: Using Blockchain to Protect Personal Data," 2015 IEEE Security and Privacy Workshops, pp. 180-184, 2015. DOI: 10.1109/SPW.2015.27.
- Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts," 2016 IEEE Symposium on Security and Privacy, pp. 839-858, 2016. DOI: 10.1109/SP.2016.55.
- K. Christidis and M. Devetsikiotis, "Blockchains and Smart Contracts for the Internet of Things," IEEE Access, vol. 4, pp. 2292-2303, 2016. DOI: 10.1109/ACCESS.2016.2566339.
- Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "MedRec: Using Blockchain for Medical Data Access and Permission Management," 2016 2nd International Conference on Open and Big Data, pp. 25-30, 2016. DOI: 10.1109/OBD.2016.11.
- X. Yue, H. Wang, D. Jin, M. Li, and W. Jiang, "Healthcare Data Gateways: Found Healthcare Intelligence on Blockchain with Novel Privacy Risk Control," Journal of Medical Systems, vol. 40, no. 10, art. 218, 2016. DOI: 10.1007/s10916-016-0574-6.
- Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends," 2017 IEEE International Congress on Big Data, pp. 557-564, 2017. DOI: 10.1109/BigDataCongress.2017.85.
- T.-T. Kuo, H.-E. Kim, and L. Ohno-Machado, "Blockchain Distributed Ledger Technologies for Biomedical and Health Care Applications,"

Journal of the American Medical Informatics Association, vol. 24, no. 6, pp. 1211-1220, 2017. DOI: 10.1093/jamia/ocx068.

Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, "Blockchain for IoT Security and Privacy: The Case Study of a Smart Home," 2017 IEEE International Conference on Pervasive Computing and Communications Workshops, pp. 618-623, 2017. DOI: 10.1109/PERCOMW.2017.7917634.

O. Novo, "Blockchain Meets IoT: An Architecture for Scalable Access Management in IoT," IEEE Internet of Things Journal, vol. 5, no. 2, pp. 1184-1195, 2018. DOI: 10.1109/JIOT.2018.2812239.

P. Zhang, J. White, D. C. Schmidt, G. Lenz, and S. T. Rosenbloom, "FHIRChain: Applying Blockchain to Securely and Scalably Share Clinical Data," Computational and Structural Biotechnology Journal, vol. 16, pp. 267-278, 2018. DOI: 10.1016/j.csbj.2018.07.004.

K. Wüst and A. Gervais, "Do You Need a Blockchain?" 2018 Crypto Valley Conference on Blockchain Technology, pp. 45-54, 2018. DOI: 10.1109/CVCBT.2018.00011.

M. Conti, E. Sandeep Kumar, C. Lal, and S. Ruj, "A Survey on Security and Privacy Issues of Bitcoin," IEEE Communications Surveys & Tutorials, vol. 20, no. 4, pp. 3416-3452, 2018. DOI: 10.1109/COMST.2018.2842460.