Blockchain for Academic Transcript Integrity

(Connecting “Blockchain Technology: Core Mechanisms…” and “Blockchain vs Distributed Databases: Dichotomy and Fusion”)

This research explores how blockchain technology can be used to ensure the integrity and verifiability of academic transcripts in university record systems. The goal is to design a prototype that hashes transcript data on a blockchain and evaluates its resistance to tampering and unauthorized modification. To understand the reasoning behind this idea, we must first look at how blockchain operates as a distributed system and how it compares with distributed databases.

---

1. Blockchain as a Distributed System

Blockchain functions as a special type of distributed system. In a normal distributed database, there is usually a **central administrator** or server that coordinates updates and ensures that all nodes hold the same data. Blockchain eliminates the need for a central authority. Instead, it uses a **consensus algorithm**, which is a set of mathematical rules that allows all nodes in the network to agree on a single, verified version of the data.

Each node in the blockchain network stores an identical copy of the ledger. When a new transaction or record is added, all nodes must validate and agree upon it before it becomes permanent. This process removes single points of failure and ensures that data remains available and trustworthy even if some nodes fail or act maliciously.

In short, blockchain equals a distributed database combined with consensus algorithms and cryptography. The consensus mechanism replaces the central coordinator, and cryptographic verification ensures that every stored record remains tamper-proof.

---

2. Consensus Mechanisms

The consensus mechanism determines how agreement is reached among network nodes. The main types are:

Proof of Work (PoW)
This mechanism was introduced in Bitcoin. Nodes, called miners, compete to solve complex mathematical puzzles. The first to solve the puzzle gets to add the next block of transactions to the chain. Because solving these puzzles requires real computing power and energy, it is extremely difficult for attackers to alter the data. PoW provides strong security but consumes vast amounts of energy and limits transaction speed.

Proof of Stake (PoS)
PoS replaces energy-intensive mining with economic staking. Validators are chosen based on how many tokens they lock as collateral. If they behave dishonestly, they lose their stake. This approach drastically reduces energy use while maintaining security, but it can lead to wealth concentration because those with larger stakes gain more influence.

Practical Byzantine Fault Tolerance (PBFT)
PBFT is used mainly in private or enterprise blockchains such as Hyperledger Fabric. It works through a voting process among a fixed number of authorized nodes, ensuring that at least two-thirds must agree on any update. PBFT provides very fast final confirmation and uses little energy, but it only scales efficiently to small or permissioned networks where participants are known.

Together, these consensus mechanisms illustrate how blockchain maintains trust and integrity across multiple nodes without needing a central server.

---

3. Role of Cryptography in Blockchain

Cryptography secures every part of blockchain operation. Hash functions convert data into unique fixed-length strings called hashes. Each block contains the hash of the previous block, which links them into an unbreakable chain. Changing even one character in a previous block would completely change its hash, exposing tampering instantly.

Public-private key cryptography ensures that only the legitimate owner of a record can authorize changes. Transactions are digitally signed using private keys and can be verified using corresponding public keys.

Merkle trees are specialized data structures that allow a system to verify whether a specific transaction exists in a block without downloading the entire blockchain. This makes verification fast and efficient even for large datasets.

By combining these cryptographic methods, blockchain ensures that stored information cannot be altered secretly and that every change is publicly verifiable.

---

4. Why Blockchain is a Distributed System

Blockchain embodies the key goals of any distributed system: consistency, availability, and fault tolerance.
It maintains consistency because all nodes eventually hold the same verified ledger. It ensures availability because multiple nodes store copies of the same data, so information is not lost even if some nodes fail. It achieves fault tolerance because the network can continue to operate correctly despite some nodes being offline or compromised.

However, blockchain introduces trade-offs. The same mechanisms that make it secure—replication, consensus, and cryptographic verification—also make it slower and less scalable than traditional databases. Privacy can also be difficult because the ledger is transparent by design. Understanding these trade-offs is essential when applying blockchain to real-world systems such as academic record management.

---

5. Distributed Databases and Blockchain: The Fusion Concept

Distributed databases and blockchains are both systems for recording and managing transactions, but they differ in their goals.
Blockchains are built for **security and trust** in untrusted environments, while distributed databases are built for **performance and scalability** in trusted organizational settings.

Universities typically use distributed databases such as MongoDB or MySQL to manage large volumes of student data. These databases handle queries and updates quickly but rely on authorized administrators, making them vulnerable to tampering or insider misuse. Blockchain, on the other hand, guarantees data immutability through decentralized consensus but has slower performance.

The most effective approach is to combine the two. This hybrid model allows databases to handle normal operations efficiently while blockchain provides verifiable proof of authenticity.

---

6. Understanding the Comparison (from Ruan et al., 2021)

Ruan and colleagues analyzed blockchains and distributed databases across four main design areas—replication, concurrency, storage, and sharding—and demonstrated how each system optimizes different goals.

Replication
Blockchains replicate entire transactions across all nodes so that every participant can verify them independently. Databases replicate only the required data fragments, improving speed but depending on trust in the central coordinator.

Concurrency
Blockchain executes transactions sequentially to keep the process deterministic and auditable. Databases allow multiple transactions to run concurrently, using concurrency control algorithms to prevent conflicts, thereby increasing throughput.

Storage
Blockchain stores all transaction history permanently in an append-only ledger, which guarantees traceability but consumes large storage space. Databases focus on storing current states with periodic pruning of logs, making them more storage-efficient.

Sharding
In databases, sharding divides data among multiple nodes based on usage patterns to enhance performance. In blockchains, shard formation also considers security—ensuring no shard contains too many malicious nodes.

7. Hybrid (Fusion) Systems

Several recent projects attempt to merge blockchain's security with database performance.

- **BlockchainDB** uses blockchain for tamper-proof storage and adds database features on top.
- **Veritas** applies cryptographic proofs to traditional databases, allowing users to verify that data has not been altered.
- **ChainifyDB** integrates blockchain ordering with relational databases to synchronize updates securely.
- **BigchainDB** combines MongoDB with a blockchain consensus layer to achieve both scalability and data integrity.

These hybrid systems show that blockchain features can strengthen database reliability without replacing their efficiency. In a university setting, this means transcript data can remain query-friendly for administrators yet cryptographically verifiable by external entities.

8. Performance Considerations

According to performance studies in the paper, distributed databases such as TiDB or etcd process tens of thousands of transactions per second, while permissioned blockchains such as Hyperledger Fabric and Quorum handle only a few thousand. Hybrid systems occupy a middle ground: they are slower than pure databases but significantly more secure.

This balance is important for academic institutions. Transcript verification does not require thousands of transactions per second, but it does require absolute trust that once data is stored, it cannot be secretly modified.

9. Application to University Transcript Systems

A practical implementation can follow three main layers:

1. **Distributed Database Layer (e.g., MongoDB)**
   Stores student profiles, course grades, and transcript data. It supports fast queries, updates, and large-scale storage across multiple university servers.
2. **Blockchain Verification Layer (e.g., Hyperledger Fabric)**
   Stores hashes of transcript records. When a transcript is issued or updated, a

cryptographic hash of the file or data entry is recorded on the blockchain.

3. **Verification Process**
   When an employer or institution needs to verify a transcript, the system recomputes the hash from the presented record and compares it with the hash stored on the blockchain. If they match, the transcript is authentic; if not, it has been tampered with.

This architecture ensures that academic records remain verifiable and resistant to unauthorized modification while maintaining the scalability and speed of existing database infrastructure.

---

10. Conclusion

Blockchain technology extends the principles of distributed systems by introducing decentralized consensus and cryptographic verification. Distributed databases, meanwhile, provide efficient, concurrent, and scalable data management. When these two are fused, they create a hybrid architecture that is both high-performance and tamper-resistant.

In the context of academic transcript management, this hybrid model ensures that student records can be updated and accessed quickly within a university's database while their authenticity and integrity are protected permanently by blockchain. Such an approach combines the transparency of decentralized systems with the practicality of enterprise-level data management, ensuring that educational credentials remain secure, verifiable, and trusted worldwide.

BLOCKCHAINS VS. DISTRIBUTED DATABASES: DICHOTOMY AND FUSION

Experimental Evaluation — What They Tested and How

The authors didn't just talk theory — they ran a **comparative experiment** on four real systems:

1. **Hyperledger Fabric (v2.2)** – a permissioned blockchain using Raft consensus
2. **Quorum (v2.2)** – an enterprise blockchain based on Ethereum, using Raft/IBFT consensus
3. **TiDB (v4.0)** – a NewSQL distributed database using Raft for replication
4. **etcd (v3.3)** – a distributed key-value store (NoSQL) using Raft consensus

These represent **both worlds**:

- Two **blockchains** (Fabric and Quorum)
- Two **distributed databases** (TiDB and etcd)

Experimental Setup

Cluster configuration:

- **96 physical nodes** in a private cluster.
- **Each node:** Intel Xeon E5 CPU, 32GB RAM, 2TB HDD.
- **Network:** 1 Gbps Ethernet connection.
- They ran multiple trials and averaged results.

Each system ran in **full replication mode**:

- Every node held a **complete copy** of the system state.
- This setup ensured a fair comparison between blockchain and distributed DBs (both had full replication).

The Data Used

They didn't use real-world business data (like bank accounts or grades) — instead, they used **standard synthetic database workloads** that simulate typical database transactions. These are designed to test performance under controlled, repeatable conditions.

The two workloads used were:

1. **YCSB – Yahoo Cloud Serving Benchmark**
    - Used widely for benchmarking NoSQL and distributed databases.
    - Generates **synthetic records** with configurable read/write ratios and key distributions.

    - In their setup:
        - **100,000 records** per test (each record was **1 KB** in size).
        - Each record contained random key–value pairs (simple data structure like `{"key": "user123", "value": "data block"}`).
    - They tested both:
        - **100% read (query-only)** workloads
        - **100% write (update-only)** workloads
2. **Smallbank Benchmark**
    - Originally designed to simulate **bank account transactions**.
    - Mimics real-world OLTP (Online Transaction Processing) — deposits, withdrawals, transfers.
    - They used **1 million records**, each representing an account, with skewed access patterns (some accounts accessed more frequently).
    - Operations included:
        - **Balance inquiry**
        - **Deposit checking**
        - **Send payment**
        - **Amalgamate accounts**
        - **Write check**
    - Each transaction typically read and updated one or two records.

These workloads are standard in distributed system research because they represent **typical small transactional operations** — reads, writes, and updates — like those in any data-driven application (banking, user data, or in your case, university records).

What Transactions Were Performed

They measured how these systems handled:

1. **Read-only transactions** → retrieving data from the ledger or database.
2. **Update (write) transactions** → modifying existing data entries.
3. **Mixed transactions** → reads and writes combined.
4. **Cross-record transactions** → accessing multiple records in one operation (for Smallbank).

Each system was configured so that:

- In **Fabric**, transactions were proposed by clients, endorsed by peers, ordered by the ordering service, and validated by all peers.
- In **Quorum**, transactions were batched into blocks and validated through Raft/IBFT consensus.
- In **TiDB**, transactions were executed with distributed SQL processing and Raft-based consistency.
- In **etcd**, transactions were simple key-value operations replicated using Raft.

What Metrics They Measured

They focused on four main performance metrics:

1. **Throughput (transactions per second – TPS)**
    - How many successful transactions the system can process per second.
2. **Latency (milliseconds per transaction)**
    - How long it takes from submitting a transaction to confirmation.
3. **Scalability (performance vs number of nodes)**
    - How the system behaves when nodes increase from 3 → 19.
4. **Storage Overhead**
    - How much extra space the system uses per record (due to consensus metadata, hashes, and logs).

Key Results

1. Throughput Findings

- For **update (write) workloads**:
  - Fabric: ~1,200 TPS
  - Quorum: ~250 TPS
  - TiDB: ~5,000 TPS
  - etcd: ~16,000 TPS
- For **read-only workloads**:
  - Fabric: ~23,000 TPS
  - TiDB: ~87,000 TPS
  - etcd: ~280,000 TPS

So, databases clearly handled higher volume.

Reason:
Databases don't have to reach global consensus for every transaction, while blockchains must ensure every node verifies and agrees on each change before confirming.

---

2. Latency Findings

- Fabric's average latency ≈ 3–4 seconds per update
- Quorum ≈ 0.5 seconds
- TiDB ≈ 80–100 milliseconds
- etcd ≈ 10–20 milliseconds

Interpretation:
In blockchains, latency spikes because transactions go through multiple stages — endorsement, ordering, and validation — plus cryptographic verification.

3. Scalability Results

They increased node count (3 → 19) and measured throughput again:

System	3 Nodes	19 Nodes	Change
Fabric	1560 TPS	528 TPS	↓ 66%
Quorum	237 TPS	219 TPS	↓ 8%
TiDB	5697 TPS	5526 TPS	~steady
etcd	19282 TPS	6076 TPS	↓ 68%

Why blockchain dropped more:
Every additional node means more signatures, more message exchanges, and slower consensus.
Databases scale more gracefully because they use leader-based replication instead of all-to-all consensus.

4. Storage Overhead

They compared how much extra storage each system uses per 1 KB record:

- **Hyperledger Fabric:** 21 KB (ledger + signatures + block metadata)
- **TiDB:** 1 KB (only data, minimal metadata)

Blockchain storage balloons because it saves the **entire transaction history** — including hashes, timestamps, and signatures — making it immutable but space-hungry.

5. Failure Models

They also compared two consensus models in Quorum:

- **Raft (Crash Fault Tolerant)** – faster, assumes trusted nodes.
- **IBFT (Byzantine Fault Tolerant)** – slower but secure against malicious nodes. Performance was similar at small scale but IBFT showed higher variance as node count increased.

This shows that **Byzantine consensus (like PBFT)** ensures higher trust but can't scale efficiently.

What All This Means

The main insight:
Blockchain systems spend most of their time ensuring *everyone agrees securely*, while databases spend most of their time *executing and optimizing transactions efficiently.*

In other words:

- Blockchain = "agreement first, execution later."
- Database = "execution first, then synchronize."

This trade-off is why blockchain is slower — but also why it's tamper-proof.

How This Relates to our Project

If we map these experiments to your academic transcript system:

- Your **distributed database (MongoDB)** will handle fast data storage and queries (like TiDB did).
- Your **blockchain layer (Hyperledger Fabric)** will record hashes of transcript updates (like Fabric did, but lighter).
- You won't need 10,000 TPS — just secure, verifiable storage for updates like "grade added" or "degree confirmed."

So, your prototype benefits from:

- Database-like performance for daily operations.
- Blockchain-like integrity for long-term record verification.

That’s the balance this paper’s experiments demonstrate — and exactly why combining both is ideal for your system.

Performance Considerations (Detailed Explanation)

Performance is one of the biggest dividing lines between **blockchains** and **distributed databases**. Both are distributed systems, but they make **very different design choices**.

These design choices affect how fast transactions are processed, how many can be handled at once (throughput), and how long it takes for each transaction to be finalized (latency).

Let’s look at what the paper *“Blockchains vs. Distributed Databases: Dichotomy and Fusion” (Ruan et al., 2021)* found — and why those results occur.

---

1. Throughput Gap — Why Databases Are Faster

The paper compared:

- **TiDB** (a NewSQL distributed database)
- **etcd** (a distributed key-value NoSQL store)
- **Hyperledger Fabric** and **Quorum** (enterprise blockchains)

Measured throughput (transactions per second):

- TiDB: ~5,000–8,000 TPS
- etcd: ~15,000–20,000 TPS
- Hyperledger Fabric: ~1,200 TPS
- Quorum: ~600 TPS

So databases are **4–10 times faster** than blockchains.

Why this happens:

1. **Consensus Overhead:**
   In databases, only a few nodes coordinate using *lightweight consensus* (like Raft or Paxos) assuming all nodes are trusted but may crash.
   In blockchains, *every node* must participate in consensus (like PBFT or PoW), even assuming some may be malicious. That creates huge messaging and verification overhead.

2. **Replication Model:**
   Databases replicate only the **data states** they need (partial replication).
   Blockchains replicate **every transaction on every node** to make sure everyone can verify the same ledger. This ensures trust, but multiplies workload.

3. **Sequential Validation:**
   Most blockchains validate transactions one by one to ensure determinism and to prevent conflicts between smart contracts.
   Databases execute transactions **concurrently**, using concurrency control (like two-phase locking or optimistic concurrency) to speed up processing.

4. **Cryptographic Verification:**
   Every blockchain transaction must be **digitally signed** and hashed into a block.
   Databases skip this step, so their transaction cycle is shorter.

---

2. Latency — Why Blockchains Feel "Slow"

Latency is the time it takes for a transaction to be confirmed.
In the experiments:

- Fabric's update latency ≈ 3–4 seconds
- Quorum's ≈ 0.5 seconds
- TiDB and etcd ≈ under 100 milliseconds

Why latency is higher in blockchains:

- **Block creation intervals:** In blockchains, transactions are grouped into blocks, and each block must go through consensus. Until that block is confirmed, the transaction remains "pending."
- **Signature verification:** Each transaction must be verified cryptographically. This adds milliseconds per transaction — which adds up when you're processing thousands.

- **Serial execution:** Each node must re-execute or validate all transactions sequentially to maintain consistency.

---

3. Scalability and Node Count

Ruan et al. also studied what happens when the number of nodes increases.

- For **blockchains**, performance dropped as nodes increased.
  Example: Fabric dropped from 1,560 TPS on 3 nodes to just 528 TPS on 19 nodes.

- For **databases**, TiDB initially scaled up to 7 nodes (≈8,000 TPS) but then plateaued due to coordination overhead.

Why this happens:

- Blockchain consensus protocols (especially PBFT) require *every node to talk to every other node* ($O(n^2)$ communication). More nodes → exponentially more messages → slower network.

- Databases scale better because only small groups of replicas need to coordinate, not the entire network.

---

4. Impact of Workload Type

The paper used two workloads:

- **YCSB (Yahoo Cloud Serving Benchmark):** simple read/write operations.

- **Smallbank:** financial-style transactions with dependencies.

Results showed that:

- Under heavy workloads or large data records, **Quorum** slowed dramatically (from ~1500 TPS to under 60 TPS).

- **Fabric** performed more consistently but had higher validation overhead.

- **TiDB** handled heavier workloads gracefully because it uses *sharding* and *parallel execution*.

Explanation:

- Blockchain systems execute each transaction in full and then re-verify it. The more complex or data-heavy the transaction, the longer it takes.
- Databases optimize queries at a low level, using indexes and caching to manage large data efficiently.

---

5. Storage and Efficiency

The paper also analyzed **storage cost**:

- Blockchains store **every transaction ever made** (append-only ledger).
- Databases store only the current state and use logs for recovery.

For example, in Fabric, a 5 KB record used over **21 KB of storage** because of the ledger and cryptographic metadata.
In TiDB, the same record stayed close to 5 KB — no historical duplication.

So, blockchain's immutability comes at the cost of extra storage.

---

6. The Hybrid "Middle Ground"

Hybrid systems — such as **ChainifyDB** or **BigchainDB** — aim to combine:

- The **speed** and **query efficiency** of databases
- The **tamper-resistance** and **auditability** of blockchains

They do this by:

- Using a normal database (like MongoDB or PostgreSQL) for data storage and queries.
- Writing only **hashes of records** (or minimal metadata) to a blockchain ledger for verification.

This setup drastically reduces consensus and replication overhead while maintaining verifiability.

---

7. What It Means for Your Academic Transcript System

Universities don’t need thousands of transactions per second. A few hundred (even less) is enough to handle student records. What they absolutely need is **data trustworthiness** — transcripts that cannot be forged, deleted, or silently changed.

Therefore:

- Using a **distributed DB (MongoDB)** ensures that student data is stored and retrieved quickly.

- Adding a **blockchain verification layer (Hyperledger Fabric)** ensures that any modification leaves a cryptographic trace.

- The overall system may not match the speed of pure databases, but it will be fast enough for university operations and much more secure.

You’re trading a small amount of performance for a massive gain in trust, transparency, and integrity — which is the right balance for this kind of system.
