Journal of Technology Informatics and Engineering (JTIE)
Vol. 5 No. 2 August 2026
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 349-379
DOI: https://doi.org/10.51903/jtie.v5i2.555

Adaptive Scalability Optimization for Blockchain-Powered Academic Credential Repositories Using Intelligent Caching and Metadata-Aware Sharding

Blessing Emmanuel Oladele*[1], Adekunle Olugbenga Ejidokun[1], Chukwuemeka Odi Agwu[1]
Email: blessing.oladele@kiu.ac.ug*, ejidokun.olugbenga@kiu.ac.ug, chukwuemeka.odi@kiu.ac.ug
[1]*Department of Computer Science. School of Mathematics and Computing. Kampala International University, Kampala, Uganda, +256*
*Corresponding Author

Abstract
Academic credential verification remains difficult for institutions because manual checks are slow, fragmented, and vulnerable to fraud. Blockchain can improve trust by anchoring credential proofs, but repeated verification requests and growing off-chain repositories can still create performance bottlenecks. This study presents an adaptive blockchain-powered academic credential repository that combines off-chain MySQL storage, Solidity-based hash anchoring, Redis verification caching, and metadata-aware sharding. Full academic records are not stored on-chain or in Redis; only credential hashes, verification responses, and related metadata are used for trust validation and performance optimization. A CodeIgniter 4 prototype was evaluated using synthetic academic credential records and controlled workloads of 1,000, 5,000, and 10,000 verification requests under fresh, mixed, and repeated access patterns. The results show that Redis caching substantially reduced repeated blockchain queries, especially under mixed and repeated workloads, while metadata-aware sharding improved repository organization and supported more targeted credential retrieval. Sepolia testnet validation confirmed smart-contract feasibility, including issuance, verification, revocation, gas use, confirmation time, and event evidence, but was treated separately from scalability testing. The findings indicate that combining blockchain trust anchoring with cache-aware verification and metadata-based repository partitioning can improve the scalability of academic credential repositories, provided that cache consistency, revocation handling, and deployment limitations are carefully managed.

Keywords: *Academic Credential Verification, Distributed Educational Repository, Intelligent Caching, Metadata-Aware Sharding, Redis, Scalability Optimization.*

I. INTRODUCTION

Academic records are produced, stored, and authenticated in a new manner with the growing digitization of higher education. Digital systems are becoming an integral part of the life of universities, playing a vital role in the management of university records, including certificates, transcripts, learner records, and other institutional documents. These systems have enhanced access and administrative effectiveness, but there are also issues of authenticity of credentials, trust, and independent verification of credentials that still pose challenges (Grech et al., 2021; Alsobhi et al., 2023; Rustemi et al., 2024). Education is relevant to a wide variety of employment, admissions, licensing, scholarship, and professional mobility decisions. These records are often used by employers, universities, regulatory agencies, and other organizations in making important decisions. Many credential verification processes, however, are still based on manual institutional correspondence, siloed databases, or paper-based processes that are slow, disjointed, and challenging to scale (Ocheja et al., 2020). With the rise of online learning and digital certification, the shortcomings of these traditional methods have come to light.

Received in May 2026; Revised in June 2026; Accepted in July 2026; Published on August 2026

The popularity of blockchain technology in recent years is due to its potential for decentralized, tamper-resistant, and verifiable record management. In the education sector, blockchain technology can be harnessed for the purpose of establishing a "block" that links a digital credential with a cryptographic proof of that credential, which can be verified independently from the continuous involvement of the institution (Grech et al., 2021; Zhao et al., 2024; Chotikakamthorn et al., 2024). There are several projects, such as Blockcerts, that have proven blockchain verification of credentials is technically viable (Sharples & Domingue, 2020).

While the benefits of blockchain-based learning repositories are significant, one of the primary challenges is their scalability. The current systems are primarily directed towards only authenticity and immutability of credentials with little attention to the long-term performance under heavy institutional workloads. However, these blockchain networks are naturally limited by their transaction throughput, the amount of storage they require, and the time needed to verify transactions particularly when a lot of verification requests are placed (Zheng et al., 2017). In a practical setting in a university, the repository can receive thousands of requests to verify credentials over a long time, especially when a scalable solution is not implemented, causing performance bottlenecks (Alsobhi et al., 2023; Gupta, 2024).

The other problem that was noted in many of the already existing systems is the repeated execution of the same blockchain verification operations. When a user requests verification from another service, each request will result in another blockchain check even if the user has previously requested verification from a different service. This will slow down the speed of verification and add unnecessary processing load. In addition, storing large educational datasets directly on-chain raises concerns relating to storage efficiency, privacy, and regulatory compliance.

To tackle these challenges, this work advocates an adaptive scalability optimization framework for academic credential repositories based on scalable intelligent caching and metadata-aware sharding techniques. The framework integrates blockchain trust anchoring, the Redis intelligent verification cache and distributed metadata-driven repository partitioning. The caching mechanism minimizes the number of blocks and interactions on the blockchain by storing and reusing the responses to previously verified credentials, and the metadata-aware sharding model enhances the organization and retrieval speed of the repositories by partitioning them into segments based on metadata.

The proposed architecture will be a hybrid one where full academic records will be stored in secure off-chain repositories, and only cryptographic proofs will be anchored on-chain. This not only minimizes storage requirements on the blockchain but also ensures the integrity and

auditability of the credentials. This study extends blockchain-based credential verification by treating the repository as both a trust system and a high-frequency verification service. Its contribution is the integration of blockchain hash anchoring, Redis-based verification caching, and metadata-aware repository partitioning in one prototype. The framework is designed for repeated verification requests, growing credential repositories, and off-chain storage of academic records.

The highlights of this study are as follows:

i. The study proposes an adaptive optimization framework for blockchain-powered academic credential repositories using workload-aware intelligent caching and metadata-aware distributed partitioning.

ii. The research proposes a Redis-based intelligent verification caching mechanism to reduce the number of verification operations on the blockchain and accelerate the verification speed.

iii. The study presents a metadata-aware sharding model that partitions the educational repositories based on the educational credential metadata to achieve better workload distribution and efficient retrieval.

The architecture is a hybrid, with sensitive academic records kept off-chain, and cryptographic proofs being securely stored on-chain.

II. LITERATURE REVIEW

The applications of blockchain in the field of higher education have been growing in prominence due to its traits of decentralized trust, immutability and credential verification. The application of blockchain in the field of certificate issuance, management of academic transcripts, learner identity management, and decentralized academic verification platforms has been explored in several studies (Grech & Camilleri, 2017). An important driver of these initiatives has been fraud on certificates and the lack of efficiency of manual verification procedures. Initial educational blockchain-based experiments, like Blockcerts, have shown that academic credentials can be issued and independently verified using blockchain technology without any permanent participation of the issuing institutions (Grech et al., 2021; Rustemi et al., 2024; Zhao et al., 2024). Likewise, Ocheja et al. (2018) suggested distributed learner record systems that can promote lifelong learning accomplishments across institutions. These studies proved the practicality of using blockchain for the verification of credentials in education.

However, even with this progress, several current blockchain educational repositories are still considered proof-of-concept with only minimal, if any, efforts being put forward to tackle the

issues of deployment at a large scale. Most systems just concentrate on authenticity of the credential and decentralized verification, and don't really optimize performance, workloads, and long-term scalability (Alsobhi et al., 2023; Esguerra et al., 2024; Gupta, 2024). Scalability is also growing in importance as the magnitude of the number of verifications in the educational system is expected to be high and at high frequencies.

Another major drawback of blockchain systems is their lack of scalability. These blockchain networks, however, suffer from high latency for reaching consensus, high computational complexity, high storage demands, and low transaction throughput (Wang et al., 2020; Zhao et al., 2024; Gupta, 2024). They are more prominent in educational settings where repositories can handle thousands of credential issuance and verification transactions over short periods, such as during an institutional audit or graduation season, or a recruitment process. One of the other drawbacks of current systems is the repetitive verification of the blockchain. In many cases, each verification request will cause a new blockchain check to be performed even if the credential has been previously verified. This method maintains verification integrity, but adds needless latency and processing overhead. As a result, under large-scale institutional workloads, it can be said that repeated blockchain interactions can cause a drop in the system's responsiveness (Kaneriya & Patel, 2023; Chotikakamthorn et al., 2024).

Some studies have suggested hybrid blockchain architectures that leverage blockchain technology mainly for trust anchoring, while full educational records are kept off chain (Ma & Fang, 2020; Kaneriya & Patel, 2023; Rustemi et al., 2024). This way, blockchain storage requirements are minimized, and privacy is enhanced. Many current hybrid architectures, however, still fail to support optimizations for adapting scalability, like intelligent coaching and distributed repository partitioning. The use of caching in distributed systems for enhancing application responsiveness and minimizing unnecessary database interactions has been employed for long time. Of particular note, however, is Redis' lightweight in-memory architecture and its fast retrieval model (Redis Labs, 2024). Caching can be especially beneficial in a blockchain setting, where it might store the results of previous validations to avoid repeated interactions with the blockchain. But the integration of smart caching in blockchain based education repositories is not sufficiently researched area.

Likewise, the technique of sharding has been extensively used in distributed database systems to enhance the scalability and workload distribution. Sharding divides a large data set into smaller, logically-related pieces, thus alleviating centralized bottlenecks and enhancing data retrieval efficiency. For educational repositories, metadata-aware sharding is a valuable approach for splitting a record based on an institution's attribute, like faculty, department, graduation year or

credential category. Although the benefits are great, there are limited studies that have explored the application of metadata-driven sharding in blockchain-based academic credential systems. The literature, therefore, shows that there are a number of gaps. Firstly, the majority of blockchain educational repositories still focus on the authenticity of credentials rather than long-term scaling and distributed optimisation. Second, there is a lack of well-studied strategies for efficient verification caching that can help minimize repeated interactions with the blockchain.

Third, the existing studies address few sharding strategies for educational repositories that are based on metadata. Lastly, many current systems are still at the conceptual stage without a strong emphasis on institutional-scale deployment readiness. The reviewed studies also show that scalability is often discussed mainly as a blockchain-infrastructure issue, while repository workload behaviour receives less attention. In university settings, credential access may be repetitive, seasonal, and shaped by metadata such as department, programme, institution, or graduation year. These access patterns justify optimization techniques that combine blockchain trust with cache reuse and metadata-aware retrieval.

Table 1: Comparative Analysis of Existing Blockchain Credential Verification Studies

Study	Area of Focus	Caching Support	Sharding Support	Scalability Observations
Grech & Camilleri (2017)	Blockchain credential verification	No	No	Focuses primarily on blockchain adoption in education rather than repository optimization.
Sharples & Domingue (2020)	Decentralized credential systems	No	No	Emphasizes credential authenticity and trust verification with limited scalability discussion.
Ocheja et al. (2018)	Distributed learner records	No	No	Addresses learner records but provides limited workload optimization mechanisms.
Ma & Fang (2020)	Hybrid blockchain repositories	No	No	Uses hybrid storage architecture but provides limited adaptive scalability support.
Proposed Study	Intelligent coaching and metadata-aware sharding	Yes	Yes	Integrates coaching, partitioning, and workload-aware optimization for repository scalability.

III. PROPOSED ADAPTIVE SCALABILITY OPTIMIZATION FRAMEWORK

The proposed framework combines blockchain trust anchoring with Redis-assisted verification caching and metadata-aware repository partitioning. Full academic records remain in off-chain repositories, while the blockchain stores only cryptographic proofs and status references. Redis is used only as a temporary verification cache, and sharding organizes credential records by metadata to support targeted retrieval. The entire architecture of the proposed adaptive scalability optimization framework is shown in Figure 1.

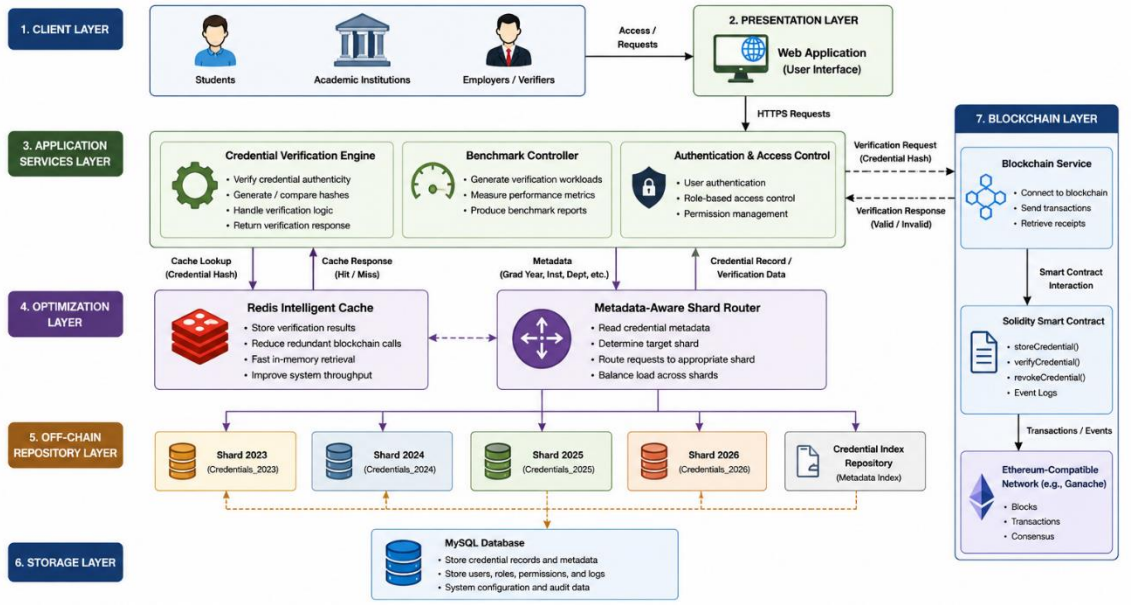

Figure 1: Overall Architecture of the Proposed Adaptive Credential Verification Framework

A. Architectural Components of the Framework

The framework is organized into five layers. The presentation layer supports institutions, credential owners, administrators, and verifiers. The application services layer handles authentication, verification routing, lifecycle operations, and API coordination. The adaptive optimization layer contains the Redis verification cache and the metadata-aware shard manager. The blockchain trust layer anchors credential hashes, status references, and revocation evidence through Solidity smart contracts. The distributed data layer stores full academic records off-chain in MySQL-based central and sharded repositories.

B. Adaptive Verification Workflow

The intelligent coaching, distributed shard routing, and blockchain trust validation are an integrated verification process. The framework does not require caching everything in its entirety because it dynamically calculates the most optimal verification path depending on the on-the-fly availability of the cache and repository metadata. The verification process begins when a verifier submits a credential identifier together with the credential hash or the data required to generate the hash. The presence of an entry in Redis is not treated as sufficient evidence that the credential is still valid. Before a cached response is reused, the system checks its expiry time, integrity information and the most recent credential status available from the blockchain event index. A stale or inconsistent entry is removed and the request continues through the normal verification path.

Journal of Technology Informatics and Engineering (JTIE)
Vol. 5 No. 2 August 2026
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 349-379
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

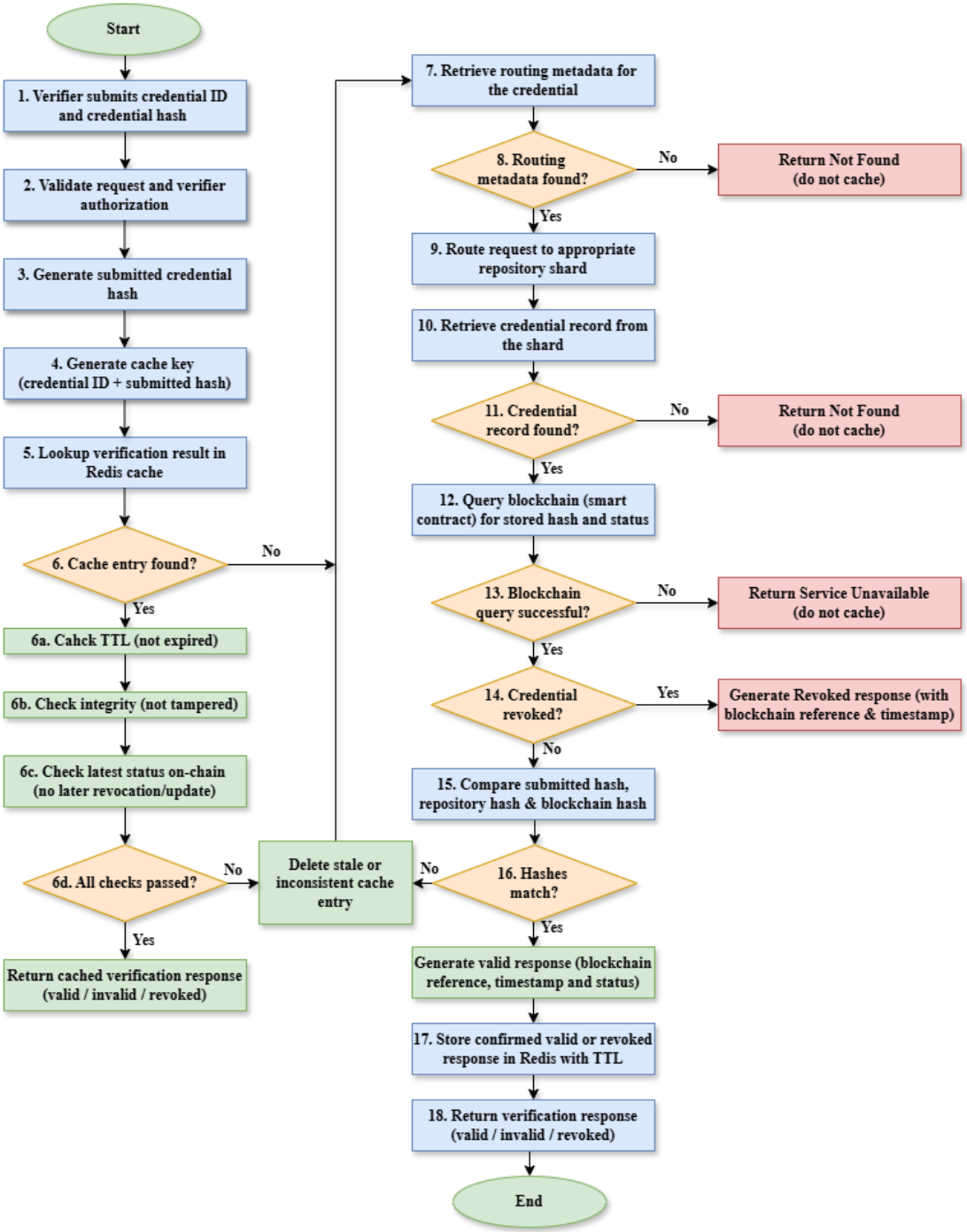

Figure 2: Adaptive Credential Verification Workflow with Redis Cache and Blockchain Validation

Where no valid cache entry is available, the system obtains the routing metadata associated with the credential and directs the request to the appropriate repository shard. The retrieved record is hashed using the same procedure used when the credential was issued. The resulting hash is compared with the hash stored through the smart contract, while the credential's revocation status is checked before a verification response is produced. Only a valid or confirmed revoked response

is stored for later reuse. Missing records, hash mismatches and temporary blockchain communication failures are not cached.

Algorithm 1: Adaptive Credential Verification and Cache-Control Algorithm

Input
Credential Verification Request, CVR, containing:
a. credential identifier;
b. submitted credential hash or credential data;
c. verifier access token.

Output
Verification Response, VR: Valid, Invalid, Revoked, Not Found, or Service Unavailable.
Algorithm Steps:
1. Receive the credential verification request.
2. Validate the request and confirm that the verifier is permitted to use the verification service.
3. Generate the submitted credential hash using Keccak-256 where credential data, rather than a hash, was supplied.
4. Generate the cache key from the credential identifier and the submitted credential hash.
5. Search Redis for an existing verification response using the cache key.
6. If a cache entry exists:
   a. Check that the entry has not expired.
   b. Check that the entry has not been altered.
   c. Check that no later credential update or revocation event has been recorded.
   d. If all three checks are successful, return the cached verification response.
   e. Otherwise, delete the stale or inconsistent cache entry and continue with Step 7.
7. Retrieve the routing metadata associated with the credential identifier.
8. If the routing metadata cannot be found, return Not Found and do not cache the response.
9. Use the routing metadata to identify the appropriate repository shard.
10. Retrieve the credential record from the selected shard.
11. If the credential record cannot be found, return Not Found and do not cache the response.
12. Generate the repository hash from the retrieved credential record using Keccak-256.
13. Query the smart contract for the stored credential hash, revocation status and blockchain reference.
14. If the blockchain query fails, return Service Unavailable and do not cache the response.
15. If the credential is marked as revoked:
    a. Generate a Revoked response.
16. Otherwise, compare the submitted hash, repository hash and blockchain hash.
17. If the three hashes do not match:
    a. Return Invalid and do not cache the response.
18. If the hashes match:
    a. Generate a Valid response.
19. Store the confirmed Valid or Revoked response in Redis using the configured time-to-live.

Return the verification response.

End

C. Intelligent Verification Caching Model

A major contributor to the loss of efficiency in credential repositories operating on blockchain is the repeated execution of the same blockchain verification operations. In most traditional solutions, each individual credential verification request requires a new request on the blockchain, even if the credential has been previously verified. The limitation is addressed by proposing an intelligent Redis-based verification caching mechanism, which dynamically stores and reuses previously validated verification responses. The caching system helps to reduce unnecessary interactions with the blockchain and significantly enhances the responsiveness of the verification

Journal of Technology Informatics and Engineering (JTIE)
Vol. 5 No. 2 August 2026
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 349-379
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

process during repeated verification tasks. Figure 3 presents the cache-consistency and invalidation workflow used to control the reuse of credential verification responses in Redis.

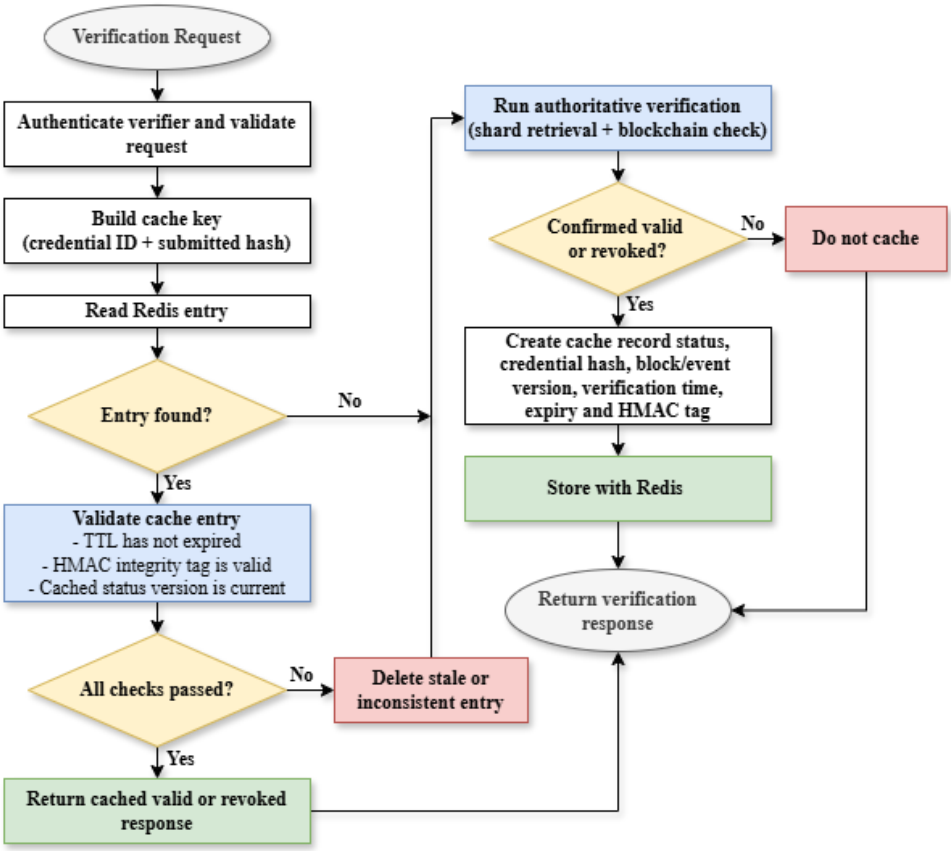

Figure 3: Redis Verification Cache Consistency and Invalidation Workflow

The cache key is generated from the credential identifier and the submitted credential hash. When a matching entry is found, the system does not return it immediately. It first confirms that the entry is still within its 300-second time-to-live, that its HMAC integrity tag is valid, and that the credential status version stored with the entry agrees with the latest version held by the status index. If any of these checks fail, the entry is removed, and the request is processed again through repository retrieval and blockchain verification. Cache invalidation is mainly event-driven. The application listener monitors credential issuance, update, and revocation events emitted by the smart contract. When a credential is updated or revoked, the status version is changed, and all Redis entries associated with the credential are removed.

A reconciliation task also runs every 60 seconds as a fallback in case a blockchain event is delayed or missed. Only responses confirmed as Valid or Revoked are stored in the cache. Invalid requests, missing credential records, and temporary blockchain communication failures are returned without being cached. Each cache record contains the credential status, credential hash, relevant block or event version, verification time, expiration time, and integrity tag. Redis

therefore serves only as a temporary performance layer, while the repository and blockchain remain the authoritative sources of credential data and status.

i. Cache Consistency and Security Controls

Redis is used only as a temporary optimization layer and not as the authoritative source of credential status. A cache entry is created only after repository retrieval and smart-contract validation. Raw certificates and personal academic records are not stored in Redis. Each cached value contains the verification decision, credential hash, blockchain reference, status version, verification time, expiry time, and HMAC-SHA-256 integrity tag. The default cache time-to-live is 300 seconds. Before a cached response is returned, the application checks expiry, HMAC integrity, and the latest credential status version. Expired, altered, or stale entries are deleted, and the request continues through repository and blockchain validation. Only confirmed Valid and Revoked responses are cached; Invalid, Not Found, and Service Unavailable responses are excluded. Revocation consistency is handled through immediate invalidation, event-based cache deletion, and reconciliation from the last processed block. Redis access is restricted to the application environment, protected by authentication and firewall controls, and treated as unavailable when the connection fails. In that case, verification bypasses Redis and continues through the authoritative path.

ii. Performance Measurement Models

Three simple measurement models were employed to make the evaluation of the proposed framework easier. These models were employed to describe the throughput of verification, cache re-use and fewer blockchain queries on the workloads evaluated.

Verification Throughput:

$$T_v = \frac{N}{L} \quad (1)$$

Tv is the verification throughput, N is the number of verification requests sent, and L is the total processing time for the verification workload.

Cache Hit Ratio:

$$CHR = \frac{H}{R} \quad (2)$$

where CHR is the cache hit ratio, H is the number of verification requests served from cache, and R is the total number of verification requests served.

BQRR stands for the Blockchain Query Reduction Rate.

$$BQRR = \frac{Q_t - Q_c}{Q_t} \times 100 \quad (3)$$

Whereas BQRR is the blockchain query reduction rate, Qt is the total number of blockchain queries that are executed without cache-assisted verification, and Qc is the number of blockchain queries that are executed after the introduction of cache-assisted verification. These models are

Journal of Technology Informatics and Engineering (JTIE)
Vol. 5 No. 2 August 2026
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 349-379
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

not meant to mathematically prove the scalability of the system, but to give a consistent interpretation of the benchmark results from the new, mixed, and repeated verification workloads. They also allow for easier comparison of the impact of cache re-use and reduction of blockchain queries for varying request sizes.

D. Metadata-Aware Sharding Architecture

With growing repository sizes, it can be problematic to manage central educational databases because they can suffer from slow retrieval speeds, indexing issues, and storage limitations. The proposed framework extends to partitioning educational records across multiple distributed repository segments by using metadata-aware sharding to enhance the scalability of distributed repository. The framework segments credential data based on metadata attributes like graduation year, institution, faculty, department, or credential type. This strategy allows for workload distribution across repository partitions and decreases retrieval complexity. The proposed framework is based on the metadata-aware sharding architecture as shown in Figure 4.

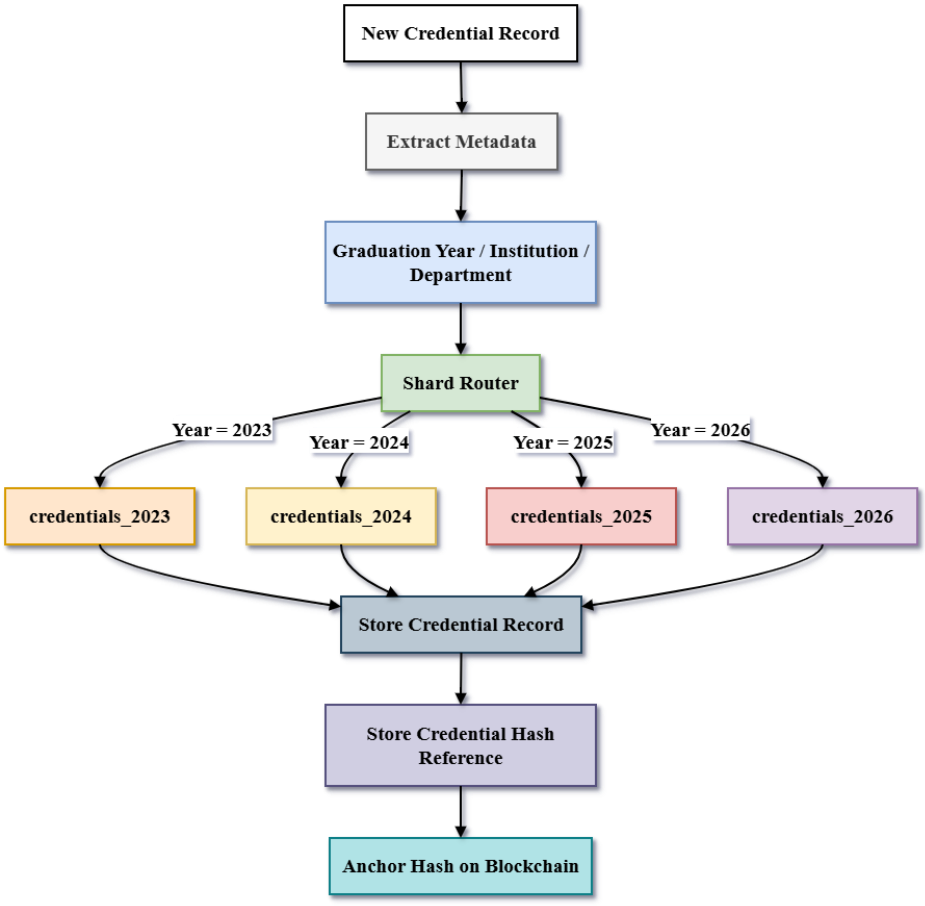

Figure 4: Metadata-Aware Sharding Architecture

The Metadata-Aware Shard Manager automatically distributes the credential records to the right repository partitions according to the extracted metadata. Requests for verification are then sent directly to the relevant shard, which reduces the amount of unnecessary distributed searches. The

partitioning strategy enhances the efficiency of localised retrieval and decreases the congestion of work in the centralized system. The framework is also designed to accommodate adaptive extension of the shards as the size of the repository and the number of institutions increase over time

i. Metadata Privacy Considerations

Although metadata attributes such as institution, faculty, programme, and graduation year are utilized for repository partitioning, the proposed framework does not expose partition structures to external users. Partition identifiers are maintained internally within the repository management layer and are inaccessible through public verification interfaces. Furthermore, only cryptographic hashes of credentials are anchored on the blockchain, while the actual credential records remain under institutional control. This approach reduces the risk of sensitive information disclosure while preserving the efficiency benefits of metadata-aware retrieval. Access to repository partitions is governed by application-level authorization mechanisms to prevent unauthorized inspection of partition structures and credential metadata.

Algorithm 2: Metadata-Aware Shard Allocation Algorithm

```
Input
  Credential Record (CR)
Output
  Shard Assignment (SA)
  Step 1:
  Extract credential metadata:
      Institution ID,
      Graduation Year,
      Department,
      Credential Type.
  Step 2:
  Use metadata attributes to generate a shard key.
  Step 3:
  Search for a matching shard partition.
  Step 4:
  If there is a matching shard:
      Use the shard object to store the credential record.
  Else:
      Create a new shard partition.
  Step 5:
  Record the number of credential records and verification requests associated with each
  shard.
  Step 6:
  Calculate the shard imbalance indicators and report a warning where the configured threshold
  is exceeded
End
```

ii. Shard Imbalance Monitoring

The partitioning strategy recognizes that repository growth may not remain evenly distributed across graduation years, institutions, or programmes. For this reason, the prototype records the number of credentials and verification requests associated with each shard. Shard imbalance is assessed using the smallest and largest shard sizes, the standard deviation and coefficient of variation of shard sizes, the largest-to-smallest shard ratio, and the percentage of records and requests handled by the busiest shard. The prototype does not automatically create new shards or

migrate existing credential records when an imbalance is detected. Instead, the measurements provide an indication of when administrative repartitioning or a more advanced rebalancing procedure may be required. Automatic shard migration and rebalancing remain extensions for a larger operational deployment.

E. Security Threat Model and Controls

The security design treats blockchain, the web application, Redis, MySQL repositories, and the RPC provider as separate trust boundaries. Blockchain provides the authoritative hash and status reference, while full academic records remain off-chain (Grech et al., 2021; Kaneriya & Patel, 2023). Off-chain tampering is detected by recomputing the credential hash and comparing it with the smart-contract value. A mismatch returns an Invalid response and is not cached. Credential issuance and revocation are restricted to authorized institutional users, while public verifiers can only check credential status. The prototype does not implement issuer-specific revocation control for multi-institution contracts; this remains a deployment requirement. Shard identifiers and routing rules are handled within the application layer and are not exposed through public verification responses. Cache-related risks are controlled through expiry checks, HMAC validation, status-version checks, revocation invalidation, and Redis bypass on failure. RPC failure produces a Service Unavailable response rather than a credential decision. The prototype did not include formal smart-contract verification, penetration testing, private-key compromise testing, or coordinated denial-of-service testing.

IV. EXPERIMENTAL EVALUATION AND PERFORMANCE ANALYSIS

The evaluation was carried out in two parts. The first part used controlled benchmark testing to examine the effect of Redis verification caching and metadata-aware sharding on response latency, tail latency, throughput, blockchain-query frequency, repository retrieval, and resource use. The second part used the Ethereum Sepolia public test network to assess smart-contract interaction, transaction cost, confirmation time, RPC behaviour, and contract-event handling. The Sepolia exercise was treated as feasibility validation and not as a public-network scalability test. The controlled benchmark used a deterministic synthetic dataset containing 10,000 academic credential records.

The records were generated from ordinary institutional credential fields, but they did not contain real student academic information. Production institutional verification logs were also not used. The dataset should therefore be understood as a synthetic, template-informed dataset rather than live institutional traffic. The same credential records were stored in a central MySQL table and in four graduation-year shards representing 2023, 2024, 2025, and 2026. Each shard contained 2,500 records during the balanced request-volume experiment. This allowed the same credential set to

be retrieved through central and sharded storage paths. Four execution modes were defined for the controlled benchmark:

i. Central No Cache: credential retrieval from the central repository with Redis bypassed;
ii. Central Cache: credential retrieval from the central repository with Redis verification caching;
iii. Sharded No Cache: metadata-aware retrieval from the appropriate repository shard with Redis bypassed; and
iv. Sharded Cache: metadata-aware retrieval combined with Redis verification caching.

The mixed and repeated workloads were evaluated using all four modes. For the fresh workload, the Central No Cache baseline was compared with the complete Sharded Cache workflow. A fresh request could not produce a valid cache hit because no verification response had previously been stored. Three controlled workload patterns were used. The fresh workload contained no prewarmed cache entries and therefore produced a 0% cache-hit ratio. The mixed workload was fixed at 60% cache hits and 40% cache misses, while the repeated workload was fixed at 90% cache hits and 10% cache misses. These ratios were kept constant for the 1,000-, 5,000-, and 10,000-request workloads.

Table 2: Experimental Dataset, Workload and Environment Configuration

Configuration Item	Description
Dataset type	Synthetic, template-informed academic credential records
Real student records	Not used
Production verification logs	Not used
Number of credential records	10,000 distinct records
Balanced shard arrangement	Four graduation-year shards: 2023, 2024, 2025 and 2026
Records per balanced shard	2,500 records
Storage baselines	Central MySQL table and metadata-aware sharded repository containing the same credential records
Request-volume workloads	1,000, 5,000 and 10,000 verification requests
Workload patterns	Fresh: 0% cache hits; Mixed: 60% cache hits; Repeated: 90% cache hits
Controlled execution modes	Central No Cache, Central Cache, Sharded No Cache and Sharded Cache
Measured trials	Five trials per experimental configuration
Request-volume load setting	25 configured JMeter threads with a 10-second ramp-up
Concurrent-access supplement	Configured thread levels of 1, 25 and 50
Backend framework	CodeIgniter 4.7.2
Programming language	PHP 8.2.12
Database	MySQL
Cache engine	Redis through Memurai
Smart-contract language	Solidity
Controlled blockchain setting	Ethereum-compatible controlled verification environment
Public blockchain validation	Ethereum Sepolia public test network, Chain ID 11155111
Load-testing tool	Apache JMeter in non-GUI mode
Web server	Apache through XAMPP
Operating environment	Windows/XAMPP controlled same-host prototype
Runtime controls	OPcache enabled; Xdebug and benchmark debug toolbar disabled
Environment qualification	Same-host prototype comparison; production deployment capacity was not established

Journal of Technology Informatics and Engineering (JTIE)
Vol. 5 No. 2 August 2026
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 349-379
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

In the cache-assisted modes, entries intended to produce cache hits were prepared before the timed benchmark began, while entries intended to produce cache misses were confirmed to be absent. Cache preparation was excluded from the reported latency and throughput. The no-cache modes bypassed Redis throughout the measured request path. Each request-volume configuration was executed in five measured trials. The reported values are therefore based on the mean of five trials rather than a single run. In total, the request-volume evaluation produced 150 measured trials and 800,000 successful credential verification requests, with no failed requests recorded.

Apache JMeter was executed in non-GUI mode on the same Windows/XAMPP computer as the repository application, and communication took place through the localhost interface. Consequently, JMeter, Apache, MySQL, and Redis shared the same processor and memory resources, while ordinary network transmission delay was not represented. The results are therefore interpreted as controlled same-host prototype measurements and not as evidence of production or multi-institution capacity. Additional experiments examined configured JMeter thread levels of 1, 25, and 50, balanced and skewed shard distributions, cache-consistency and security controls, and actual Ethereum Sepolia transaction and read operations. The configuration used for the evaluation is summarized in Table 2.

A. Evaluation Metrics

The evaluation considered response performance, cache behaviour, blockchain activity, repository retrieval, shard distribution, reliability and resource use. The main measurements were:

i. mean, median, p95 and p99 verification latency;
ii. verification throughput in requests per second;
iii. successful and failed request counts and error rate;
iv. cache hits, cache misses and cache-hit ratio;
v. blockchain-query count and blockchain-query reduction rate;
vi. repository retrieval time and shard-routing overhead;
vii. shard size, request distribution, coefficient of variation, largest-to-smallest shard ratio and busiest-shard percentage;
viii. host CPU and memory use, Redis memory use and MySQL connection activity; and
ix. Sepolia gas use, transaction fee, confirmation time, RPC latency, RPC failure rate and contract-event processing.

Mean latency, p95 latency, p99 latency and throughput were calculated separately for each measured trial. The reported configuration values were then obtained from five independent trials. A 95% Student's t confidence interval was calculated from the five trial-level values. For direct comparisons between execution modes, corresponding trial numbers were paired and a two-sided paired Student's t-test was used. A result was not described as statistically significant unless the

calculated p-value was below 0.05. The Redis state was prepared separately for each workload. The cache was empty for the fresh workload, while the intended hit entries for the mixed and repeated workloads were prewarmed outside the measured period. A unique cache namespace was used for each trial to prevent one trial from affecting another. Comparable execution modes used the same deterministic credential request sequence.

B. Fresh Verification Workload Evaluation

. The fresh workload represented first-time credential verification requests for which no reusable Redis entry existed before the measured run. The purpose of this test was to establish the cost of the complete authoritative verification path before cache reuse became possible. Central No Cache was used as the conventional baseline, while Sharded Cache represented the complete proposed workflow. In both modes, every request required repository retrieval and blockchain validation. The Sharded Cache mode also performed cache lookup, metadata-based shard routing and storage of the confirmed response for possible reuse in a later request. The fresh workload produced no cache hits because no credential was prewarmed. All valid requests therefore followed the authoritative repository and blockchain path. As expected, blockchain-query reduction was 0% in both compared modes. The result provides the baseline for later mixed and repeated workloads, where cache reuse becomes possible.

Table 3: Fresh Verification Workload Performance (Mean of Five Trials)

Requests	Execution mode	Mean latency, ms (95% CI)	p95 latency (ms)	p99 latency (ms)	Throughput, requests/s (95% CI)	Blockchain queries
1,000	Central No Cache	360.33 (291.54-429.12)	612.60	954.40	42.44 (37.73-47.14)	1,000
1,000	Sharded Cache	399.53 (317.50-481.56)	672.60	960.60	40.60 (35.41-45.78)	1,000
5,000	Central No Cache	483.93 (461.63-506.23)	769.00	991.20	46.56 (44.36-48.77)	5,000
5,000	Sharded Cache	524.06 (484.86-563.27)	791.40	1,007.80	44.21 (41.25-47.16)	5,000
10,000	Central No Cache	429.51 (402.39-456.63)	611.60	757.60	55.27 (51.91-58.63)	10,000
10,000	Sharded Cache	482.37 (421.76-542.97)	704.20	878.40	49.73 (44.11-55.35)	10,000

C. *Mixed Verification Workload Evaluation*

The mixed workload represented a combination of previously verified credentials and credentials requiring first-time validation. The request composition was fixed at 60% intended cache hits and 40% cache misses for all three workload sizes. This produced 600 hits and 400 misses at 1,000 requests, 3,000 hits and 2,000 misses at 5,000 requests, and 6,000 hits and 4,000 misses at 10,000 requests. All four execution modes were evaluated so that the effects of caching and repository partitioning could be examined separately. The no-cache modes bypassed Redis and therefore

Journal of Technology Informatics and Engineering (JTIE)
Vol. 5 No. 2 August 2026
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 349-379
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

performed one blockchain query for every successful verification request. In the cache-assisted modes, a valid cache hit did not require another blockchain query, while every cache miss followed the repository and blockchain verification path. The mixed workload maintained the intended 60% cache-hit and 40% cache-miss composition. Cache-enabled modes reduced blockchain queries by 60%, although latency and throughput gains varied by request volume and execution mode. This shows that caching reliably reduced blockchain dependence, but its end-to-end performance benefit depended on cache-access overhead and host resource conditions.

Table 4: Mixed Verification Workload Performance (Mean of Five Trials)

Requests	Execution mode	Cache hits / misses	Blockchain queries	Mean latency, ms (95% CI)	p95 / p99 latency (ms)	Throughput, requests/s (95% CI)
1,000	Central No Cache	Bypassed	1,000	248.19 (238.34–258.03)	409.40 / 486.00	52.88 (51.22–54.54)
1,000	Central Cache	600 / 400	400	279.84 (254.18-305.50)	474.80 / 579.40	49.01 (46.38–51.63)
1,000	Sharded No Cache	Bypassed	1,000	262.68 (238.17–287.19)	427.60 / 492.60	50.46 (48.64–52.27)
1,000	Sharded Cache	600 / 400	400	284.94 (257.98–311.90)	489.00 / 597.00	49.05 (46.45–51.65)
5,000	Central No Cache	Bypassed	5,000	398.45 (390.61–406.30)	565.20 / 633.60	56.06 (55.22–56.89)
5,000	Central Cache	3,000 / 2,000	2,000	425.29 (414.45–436.13)	601.40 / 710.00	52.54 (52.04–53.05)
5,000	Sharded No Cache	Bypassed	5,000	409.23 (395.13–423.33)	581.60 / 692.40	55.05 (53.40–56.70)
5,000	Sharded Cache	3,000 / 2,000	2,000	425.14 (404.75–445.52)	606.80 / 714.80	53.22 (51.26–55.18)
10,000	Central No Cache	Bypassed	10,000	411.11 (388.70–433.52)	566.60 / 698.00	57.71 (54.98–60.43)
10,000	Central Cache	6,000 / 4,000	4,000	428.30 (411.52–445.08)	587.20 / 671.80	55.31 (53.48–57.14)
10,000	Sharded No Cache	Bypassed	10,000	404.91 (398.76–411.05)	550.00 / 673.20	58.40 (57.43–59.37)
10,000	Sharded Cache	6,000 / 4,000	4,000	420.56 (404.49–436.62)	579.00 / 658.00	56.21 (54.20–58.22)

D. Repeated Verification Workload Evaluation

The repeated workload represented a high-reuse verification condition in which most requests referred to credentials that had already been verified. The request composition was fixed at 90% intended cache hits and 10% cache misses for all three workload sizes. This produced 900 hits and 100 misses at 1,000 requests, 4,500 hits and 500 misses at 5,000 requests, and 9,000 hits and 1,000 misses at 10,000 requests. All four execution modes were evaluated. The two no-cache modes bypassed Redis and performed one blockchain query for every successful request. In the cache-assisted modes, only the 10% cache misses continued through repository retrieval and blockchain validation. The repeated workload therefore examined whether a high level of

response reuse could reduce blockchain activity and improve end-to-end verification performance.

Table 5: Repeated Verification Workload Performance (Mean of Five Trials)

Requests	Execution mode	Cache hits / misses	Blockchain queries	Mean latency, ms (95% CI)	p95 / p99 latency (ms)	Throughput, requests/s (95% CI)
1,000	Central No Cache	Bypassed	1,000	280.76 (246.63–314.88)	457.00 / 516.60	48.94 (45.98–51.91)
1,000	Central Cache	900 / 100	100	287.70 (231.04–344.35)	459.60 / 525.60	49.38 (44.98–53.78)
1,000	Sharded No Cache	Bypassed	1,000	262.45 (230.23–294.67)	411.40 / 475.60	50.84 (48.49–53.19)
1,000	Sharded Cache	900 / 100	100	279.55 (234.87–324.23)	442.60 / 503.00	50.03 (45.56–54.49)
5,000	Central No Cache	Bypassed	5,000	378.89 (316.13–441.65)	507.40 / 605.00	59.28 (52.07–66.50)
5,000	Central Cache	4,500 / 500	500	415.95 (323.49–508.42)	559.80 / 731.20	54.77 (44.97–64.58)
5,000	Sharded No Cache	Bypassed	5,000	402.12 (346.18–458.06)	554.40 / 711.40	56.58 (50.66–62.50)
5,000	Sharded Cache	4,500 / 500	500	447.23 (291.15–603.32)	607.00 / 746.40	52.75 (39.96–65.53)
10,000	Central No Cache	Bypassed	10,000	425.07 (353.65–496.48)	563.20 / 678.60	56.28 (48.43–64.13)
10,000	Central Cache	9,000 / 1,000	1,000	442.89 (378.81–506.96)	566.80 / 699.60	53.96 (46.66–61.26)
10,000	Sharded No Cache	Bypassed	10,000	384.99 (357.55–412.42)	496.00 / 582.20	61.33 (57.49–65.17)
10,000	Sharded Cache	9,000 / 1,000	1,000	441.21 (411.86–470.57)	550.40 / 626.00	53.89 (50.57–57.21)

The repeated workload maintained the intended 90% cache-hit and 10% cache-miss composition. Cache-enabled modes therefore reduced blockchain queries by 90%. The results show stronger benefit under repeated access than under the mixed workload, confirming that cache reuse is most useful where the same credentials are verified frequently.

Table 6: Blockchain Query Reduction Across Verification Workloads

Workload	Requests	No-cache blockchain queries	Cache hits / misses	Cache-assisted blockchain queries	Query reduction (%)
Fresh	1,000	1,000	0 / 1,000	1,000	0
Fresh	5,000	5,000	0 / 5,000	5,000	0
Fresh	10,000	10,000	0 / 10,000	10,000	0
Mixed	1,000	1,000	600 / 400	400	60
Mixed	5,000	5,000	3,000 / 2,000	2,000	60
Mixed	10,000	10,000	6,000 / 4,000	4,000	60
Repeated	1,000	1,000	900 / 100	100	90
Repeated	5,000	5,000	4,500 / 500	500	90
Repeated	10,000	10,000	9,000 / 1,000	1,000	90

E. *Blockchain Query Reduction Analysis*

Table 6 applies the blockchain query reduction model presented in Equation (3). The no-cache query count was used as the reference because both no-cache modes bypassed Redis and made

Journal of Technology Informatics and Engineering (JTIE)
Vol. 5 No. 2 August 2026
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 349-379
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

one blockchain verification call for every successful request. In the cache-assisted modes, only requests that were not served from Redis continued to the blockchain verification layer. Table 6 confirms that blockchain-query reduction followed the configured workload composition. Fresh workloads produced no reduction, mixed workloads produced 60% reduction, and repeated workloads produced 90% reduction in cache-enabled modes. These values support the cache mechanism's main contribution: reducing repeated authoritative blockchain checks.

These results provide clear evidence that intelligent caching reduced repeated calls to the blockchain verification layer. However, query reduction should not be treated as automatic evidence of lower end-to-end latency. As shown in Sections 4.3 and 4.4, Redis lookup, integrity checking and response handling also introduced processing overhead. The main contribution demonstrated in Table 6 is therefore the consistent reduction in blockchain-query frequency, while the latency and throughput effects remained dependent on the workload and execution environment.

F. Effect of Metadata-Aware Sharding

To isolate the contribution of metadata-aware sharding from that of Redis caching, the sharding experiment compared the Central No Cache and Sharded No Cache execution modes. Redis was bypassed in both modes, while the same credential records and request sequences were used. Repository retrieval time was treated as the primary measurement, while end-to-end verification latency and throughput were retained as secondary measurements. The experiment used a repository containing 10,000 credential records divided into four graduation-year shards. Two distribution conditions were examined. Under the balanced condition, each shard contained 2,500 records and received 250 of the 1,000 verification requests.

Table 7: Central and Sharded Retrieval Under Balanced and Skewed Distributions

Distribution	Execution mode	Mean repository retrieval, ms (95% CI)	Mean routing overhead (ms)	Mean latency, ms (95% CI)	Throughput, requests/s (95% CI)
Balanced	Central No Cache	2.597 (2.276–2.918)	0.00000	409.49 (354.36–464.62)	37.67 (33.42–41.91)
Balanced	Sharded No Cache	2.855 (2.651–3.059)	0.00194	409.75 (389.18–430.31)	37.77 (37.04–38.50)
Skewed	Central No Cache	3.128 (2.256–4.000)	0.00000	427.94 (333.09–522.79)	35.72 (29.10–42.34)
Skewed	Sharded No Cache	3.168 (2.818–3.519)	0.00202	421.09 (364.92–477.26)	37.20 (33.29–41.11)

Under the skewed condition, the four shards contained 7,000, 1,000, 1,000, and 1,000 records, respectively, while the request distribution was 700, 100, 100, and 100. Each storage mode and distribution was evaluated in five measured trials using 25 configured JMeter threads. The balanced distribution had a shard-size coefficient of variation of 0 and a largest-to-smallest shard ratio of 1:1. The skewed distribution had a coefficient of variation of 1.2 and a largest-to-smallest

ratio of 7:1. The busiest shard accounted for 25% of the records and requests in the balanced condition and 70% in the skewed condition. Values are means of five measured trials. Values in parentheses are 95% Student's t confidence intervals. Each trial contained 1,000 requests and was completed without a retrieval error. Central No Cache did not require shard routing and therefore recorded zero routing overhead. Under the balanced distribution, mean repository retrieval time was 2.597 ms for the central repository and 2.855 ms for the sharded repository. Sharded retrieval was therefore approximately 10.0% slower at the tested repository size. However, the end-to-end results were almost identical. Mean latency was 409.49 ms for Central No Cache and 409.75 ms for Sharded No Cache, while throughput was 37.67 and 37.77 requests per second, respectively.

Under the skewed distribution, mean repository retrieval time was 3.128 ms for the central repository and 3.168 ms for the sharded repository. This represented an observed retrieval difference of approximately 1.3%. The sharded workflow nevertheless recorded a slightly lower end-to-end mean latency of 421.09 ms, compared with 427.94 ms for the central workflow. Its throughput was also higher at 37.20 requests per second, compared with 35.72 requests per second. The shard router introduced very little processing overhead. Mean routing overhead was 0.00194 ms under the balanced distribution and 0.00202 ms under the skewed distribution. The mean p95 routing overhead was approximately 0.00291 ms in both conditions. No routing or retrieval error was recorded during the experiment.

Uneven distribution affected both storage modes. When the workload changed from balanced to skewed, central repository retrieval increased from 2.597 to 3.128 ms, while sharded retrieval increased from 2.855 to 3.168 ms. The skewed condition therefore illustrates the effect of directing 70% of the repository records and requests to a single shard. This supports the need to monitor shard size and request concentration as the repository grows. The results do not show that metadata-aware sharding consistently reduced database retrieval time at the tested repository size. Its clearer contribution was the structured distribution and routing of credential records with negligible routing overhead. The framework also remained operational under the deliberately skewed 70:10:10:10 distribution. Automatic record migration or shard rebalancing was not performed in this experiment; the imbalance indicators were used only to identify conditions that may require later administrative repartitioning.

G. Concurrent-User Stress Evaluation

A supplementary concurrency experiment examined how the conventional verification path and the complete proposed workflow behaved as the number of configured JMeter threads increased. The experiment used the repeated 1,000-request workload, consisting of 90% intended cache hits and 10% cache misses. Central No Cache represented the conventional baseline, while Sharded Cache represented the complete proposed workflow. We examined JMeter thread levels of 1, 25,

Journal of Technology Informatics and Engineering (JTIE)
Vol. 5 No. 2 August 2026
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 349-379
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

and 50. Each mode and thread level was represented by five measured trials. The 25-thread results were taken from the repeated request-volume experiment because the same dataset, workload composition, and execution modes had already been tested under that condition. The 1-thread and 50-thread conditions were subsequently executed using the same 10,000-record balanced dataset. The configured thread count did not always equal the maximum number of active threads observed in the raw JMeter records. The one-thread tests reached one active thread, the configured 25-thread tests reached maximum active-thread counts ranging from 21 to 23, and the configured 50-thread tests reached maximum counts ranging from 41 to 47. The results are therefore reported using both the configured and observed thread values.

Table 8: Concurrent-User Stress Evaluation Results (Mean of Five Trials)

Configured JMeter threads	Observed maximum active threads	Execution mode	Mean latency, ms (95% CI)	p95 / p99 latency (ms)	Throughput, requests/s (95% CI)	Mean host CPU / memory
1	1	Central No Cache	54.79 (53.68–55.89)	72.80 / 89.20	18.03 (17.67–18.40)	73.81% / 10.77 GB
1	1	Sharded Cache	55.69 (54.22–57.17)	73.80 / 89.40	17.75 (17.29–18.22)	73.80% / 10.71 GB
25	21–23	Central No Cache	280.76 (246.63–314.88)	457.00 / 516.60	48.94 (45.98–51.91)	97.28% / 9.50 GB
25	21–23	Sharded Cache	279.55 (234.87–324.24)	442.60 / 503.00	50.03 (45.55–54.50)	96.24% / 9.29 GB
50	41–46	Central No Cache	653.38 (615.68–691.08)	934.20 / 5,197.80	40.24 (38.93–41.54)	95.27% / 11.59 GB
50	46–47	Sharded Cache	776.98 (746.30–807.65)	1,195.60 / 1,465.80	40.01 (38.64–41.38)	94.61% / 11.57 GB

Values are means of five measured trials. Values in parentheses are 95% Student's t confidence intervals. Each trial contained 1,000 verification requests and recorded no failed requests. Every Sharded Cache trial produced 900 cache hits, 100 cache misses, and 100 blockchain queries, representing a 90% blockchain-query reduction. Central No Cache bypassed Redis and made 1,000 blockchain queries per trial. Host CPU and memory values include Apache, MySQL, Redis/Memurai, and JMeter because the load generator and application ran on the same computer. All 30 trials completed successfully without a failed request. The Sharded Cache mode maintained the intended 90% cache-hit ratio at every configured thread level. Its blockchain-query count remained at 100 per trial, compared with 1,000 queries in the Central No Cache mode. The query-reduction benefit therefore remained stable as the configured concurrency increased.

At the one-thread setting, the two workflows produced similar results. Central No Cache recorded a mean latency of 54.79 ms and a throughput of 18.03 requests per second, while Sharded Cache recorded 55.69 ms and 17.75 requests per second. The proposed workflow was approximately 1.7% slower in mean latency and 1.6% lower in throughput. The p95 and p99 values were also

almost identical. At the configured 25-thread level, the proposed workflow recorded a mean latency of 279.55 ms, compared with 280.76 ms for the baseline. Its p95 latency decreased from 457.00 to 442.60 ms, while p99 latency decreased from 516.60 to 503.00 ms. Throughput increased from 48.94 to 50.03 requests per second. These represent observed improvements of 0.4% in mean latency, 3.2% in p95 latency, 2.6% in p99 latency and 2.2% in throughput. The confidence intervals overlap, so these results are presented as observed differences rather than statistically established improvements.

At the configured 50-thread level, Sharded Cache recorded a higher mean latency of 776.98 ms, compared with 653.38 ms for Central No Cache. Its p95 latency was also higher at 1,195.60 ms, compared with 934.20 ms. Throughput remained almost the same at 40.01 and 40.24 requests per second respectively. However, the p99 latency for Sharded Cache was 1,465.80 ms, while the central baseline recorded a pronounced tail value of 5,197.80 ms. The proposed workflow therefore reduced the extreme p99 delay, although it did not improve mean or p95 latency under the 50-thread condition. Host CPU use increased from approximately 74% at the one-thread setting to between 94.6% and 97.3% under the configured 25- and 50-thread settings. Peak CPU reached 100% in every 25- and 50-thread configuration. This indicates that the same-host computer approached processor saturation as the load increased. The latency measured at the higher thread levels therefore reflects both application processing and competition between JMeter, Apache, MySQL, and Redis for the same host resources.

Overall, the concurrent-user results demonstrate that the prototype continued to process all requests successfully at the three configured thread levels and maintained the expected 90% reduction in blockchain queries. The complete workflow produced small performance gains at the moderate 25-thread setting, but these gains did not continue at the higher 50-thread setting. The findings therefore support prototype-level concurrent operation and query reduction, but not a claim of unrestricted production capacity. The concurrency experiment was conducted through the Windows localhost interface. Normal local-area or Internet network delay was therefore not included, and the application and load generator shared the same processor and memory. The results should be interpreted as a controlled comparison of the two verification workflows under increasing same-host load rather than as distributed or multi-institution user-capacity measurements.

Where applicable, describe the ethical approval obtained, informed consent procedures, and how participant confidentiality and data security were maintained. Each subsection should be written with enough detail to enable another researcher to replicate the study accurately. The overall procedure does not need to be described in full; it is necessary to provide a reference (eg, F-test formula, t-test, etc.). It is required to present the test findings and their interpretation to

demonstrate the validity and reliability of the research instruments. The symbols on the model are described in sentences.

In addition, if the Method section includes figures, tables, flowcharts, frameworks, diagrams, or mathematical formulas, several standards must be followed to maintain consistency and clarity. All visual elements must be explicitly mentioned within the main text and integrated seamlessly into the narrative. Figures, tables, and formulas must be numbered sequentially based on the order of their appearance (e.g., Figure 1, Table 1, Equation 1), and each must be accompanied by a clear and descriptive title or caption. Images and diagrams must have a minimum resolution of 300 dpi to ensure high-quality reproduction in publication. Mathematical formulas must be properly formatted and each variable or symbol used within the equations must be clearly defined immediately before or after its first appearance. All visual representations must serve to enhance the reader's understanding of the research design and procedures.

Table 9: Functional Validation of Cache Consistency and Security Controls

Test case	Expected behaviour	Observed result	Outcome
Valid cache hit	Accept a valid and correctly signed cached response	Cache status was HIT, and the stored Valid response was accepted	Pass
Expired cache entry	Reject and remove an expired response	The expired entry was unavailable to verification, and the request returned a cache MISS	Pass
Altered cached payload	Reject a cached value modified after it was signed	The altered entry returned INVALID_HMAC	Pass
HMAC failure	Prevent use of a response with an invalid integrity tag	The response returned INVALID_HMAC and was not accepted	Pass
Status-version mismatch	Reject an entry created under an earlier credential status	The entry returned STATUS_STALE and was directed to the authoritative verification path	Pass
Immediate revocation invalidation	Remove the related cache entry immediately after revocation	One related entry was deleted, and the next lookup returned MISS	Pass
Event-based invalidation	Remove a cached response after a credential-status event	The affected Redis entry was deleted by the event-handling path	Pass
Reconciliation recovery	Recover a missed revocation event from the last processed block	Processing advanced from block 11,284,607 to 11,284,608, and the affected cache entry was deleted	Pass
Redis unavailability	Bypass Redis and continue through the authoritative path without caching a failure	Cache status was BYPASSED; authoritative verification was attempted and no cache write occurred	Pass
Invalid response handling	Do not store an Invalid verification response	The cache-store operation returned false and no cache entry was created	Pass
Not Found response handling	Do not store a missing-record response	The cache-store operation returned false and no cache entry was created	Pass
Service Unavailable handling	Do not cache an unavailable blockchain or RPC response	The cache-store operation returned false and no cache entry was created	Pass
Unauthorized request	Reject access to a protected benchmark or administrative operation	The unauthenticated request returned HTTP 401 Unauthorized and made no cache write	Pass
Issuer-specific revocation	Prevent one authorized institution from revoking another institution's credential	Issuer-specific revocation was not implemented in the current prototype	Not applicable

H. Cache Consistency and Security-Control Validation

The purpose of the test was to determine whether Redis reuse could be prevented from returning expired, altered, stale or unauthorized verification responses. The suite used controlled test credentials and did not form part of the JMeter performance workload. Fourteen functional cases were examined. Thirteen produced the expected result, while issuer-specific revocation control was recorded as not applicable because it was not implemented in the current prototype. The operational cache time-to-live remained 300 seconds. A shorter test-only expiry period was used for the expiration case so that the control could be verified without changing the operational setting. The HMAC secret was not written to the exported evidence.

"Pass" means that the observed functional behaviour agreed with the expected control. The expiration case used a short test-only TTL, while the operational cache TTL remained 300 seconds. The status-version and Redis-unavailability cases confirmed stale-entry rejection and initiation of the authoritative fallback path; the functional records did not capture the final blockchain verification response. Issuer-specific revocation was not treated as a passed control because it was not implemented. The functional controls show that Redis was not treated as an independent source of credential truth. Valid signed entries were accepted, while expired, altered, stale, invalid, missing, and unavailable responses were rejected or excluded from caching. Revocation invalidation, event-based deletion, reconciliation recovery, Redis bypass, and unauthorized-access rejection also behaved as expected. Issuer-specific revocation was not implemented and was therefore reported as not applicable rather than as a passed control. These tests support cache-consistency behaviour, but they do not constitute a formal security audit.

I. Public Blockchain Validation Using Ethereum Sepolia

To complement the controlled scalability experiments, the proposed framework was deployed and validated on the Ethereum Sepolia public test network. The objective of this validation was not to perform large-scale scalability benchmarking, but rather to assess the practical feasibility of operating the proposed architecture under real blockchain communication conditions. Unlike the simulated environment used for workload evaluation, the Sepolia test network introduces realistic factors such as remote node communication, network latency, transaction propagation delays, and smart contract interaction overhead. The smart contract was successfully deployed on the Ethereum Sepolia public test network and integrated with the cloud-hosted repository prototype. Connectivity tests confirmed successful communication between the application and the deployed contract through a remote RPC endpoint. The validation process verified contract accessibility, blockchain write capability, contract readability, and network responsiveness.

The validation demonstrated that the proposed framework can operate beyond controlled simulation environments while maintaining successful credential verification functionality. The observed network latency was significantly higher than the latency measured in the controlled

benchmark environment, reflecting the practical communication overhead associated with public blockchain infrastructures. However, the architecture remained functional and capable of supporting credential verification operations. The Sepolia validation results should be interpreted as a feasibility assessment rather than a scalability benchmark. Large-scale workload testing was intentionally retained within the controlled experimental environment to ensure repeatability and eliminate variability introduced by public network conditions, RPC provider limitations, and fluctuating blockchain activity.

Large-scale scalability experiments were intentionally conducted within the controlled benchmark environment rather than on the public Sepolia network. Public blockchain test networks are influenced by factors outside the control of the researcher, including RPC provider rate limits, variable transaction propagation times, temporary network congestion, and fluctuating node response behaviour. Conducting scalability measurements under these conditions could introduce variability unrelated to the proposed optimization techniques. Consequently, the controlled environment was used to evaluate scalability, while the Sepolia deployment was used to validate practical blockchain integration and operational feasibility.

Table 10: Ethereum Sepolia Public Testnet Validation Results

Metric	Result
Network	Ethereum Sepolia Test Network
Chain ID	11155111
Contract Deployment	Successful
Contract Readability	Successful
Write Operations	Enabled
Contract Address Configured	Yes
RPC Connectivity	Successful
Average RPC Latency	528.75 ms
Blockchain Mode	Sepolia Public Testnet

The successful deployment and validation of the proposed framework on the Ethereum Sepolia public test network provide additional evidence that the architecture can operate in a realistic blockchain environment. Although the controlled benchmark experiments remain the primary basis for scalability evaluation, the Sepolia validation confirms the practical deployability of the framework and demonstrates that the proposed caching and partitioning mechanisms can coexist with real blockchain infrastructure.

V. DISCUSSION

A. Verification Workload Behavior

The evaluation outcomes showed that there were significant differences in system behavior for each of the different verification workloads. The fresh verification workloads resulted in the lowest throughput and highest latency because all requests needed to validate the blockchain and integrity of credentials directly. Such actions are indicative of real-world institutional situations where new credentials are first authenticated. The mixed verification workloads, in contrast,

showed improved performance because some verification responses were reused in part during the verification process. The more repeated credential requests that were made, the fewer interactions the blockchain had. The most overall throughput was achieved by repeated workload since many verification requests referred to previously validated credentials. The results indicated that in an environment where access to the repository is repetitive, there are natural advantages to the use of adaptive verification mechanisms in educational credential repositories.

B. The impact of intelligent verification caching

Of great importance in the evaluation was the impact that intelligent cache reuse had on the efficiency of verification. While there was extra overhead by requiring the population of the cache and metadata synchronization during the initial verification operations, later verification operations saw a significant decrease in blockchain dependence. This behaviour became more pronounced under repeatedly and mixedly verified workloads where previously tested answers could be directly fetched from cache memory. Based on the results, it is found that intelligent caching is well suited for the institutional environment where credentials are accessed again and again in the process of admission screening, graduate verification, processing of scholarships, and background checks for the candidates while they are going for their jobs.

The framework aims to reduce redundant credential response validation by reusing previously validated responses controlled by the blockchain, as opposed to repeatedly performing the same set of operations. The findings obtained from the controlled workload experiments were further supported by the Ethereum Sepolia public testnet validation. Although public blockchain interactions introduced substantially higher communication latency than the hosted benchmark environment, successful contract interaction and verification operations confirmed the practical applicability of the proposed caching strategy in realistic blockchain deployment scenarios.

C. Effect of Metadata-Aware Sharding

The evaluation showed that intelligent caching produced the largest immediate performance gains, while metadata-aware sharding contributed primarily to repository organization and distributed retrieval efficiency. However, partition-aware repository organization enhanced the distributed retrieval efficiency, especially when the repository workload was greater. The framework allowed credential records to be organized together based on metadata properties like graduation year, institutional type, etc., which decreased reliance on a centralized repository in credential lookup operations. Additionally, the evaluation revealed that direct throughput benefits due to sharding were dependent on workload size. For cache-assisted verification, the results showed a slight decrease in the immediate performance boost seen with the shard coordination overhead and limitations on resources in the hosted environment.

Despite this, the distributed repository organization still resulted in better retrieval organization and workload distribution behaviour with increasing size of the repository. The Sepolia validation further demonstrated that repository partitioning can be integrated with blockchain-supported verification without affecting smart contract accessibility. While the public testnet experiments were not intended to evaluate large-scale partitioning performance, they confirmed the operational compatibility of the partitioning architecture with real blockchain infrastructure.

D. Combined Effect of Caching and Partitioning

The combination of controlled scalability evaluation and public blockchain validation provides stronger evidence of practical applicability than either approach alone. The benchmark experiments quantified the performance benefits of caching and partitioning under controlled conditions, while the Sepolia validation demonstrated that the proposed framework can interact successfully with a real blockchain environment.

E. Implications for Institutional Deployment

Based on the results of this study, it is possible to propose that the use of adaptive cache-assisted verification frameworks can significantly enhance the scalability of blockchain-based academic credential repositories in an institutional environment. Often, the educational repositories receive repetitive verification requests from various departments/agencies and from various verification exercises, using the same identity data. In these cases, minimizing repeated blockchain processing becomes increasingly relevant to ensure a satisfactory response to the verification. The study also confirmed that the costs of deploying a public blockchain environment for a prototype implementation of institutional verification systems are not necessarily prohibitive to realize measurable scalability benefits.

Institutions can enhance validation efficiency without compromising credential integrity or audibility through a combination of controlled blockchain validation, intelligent cache reuse, and distributed repository organization. The successful deployment of the framework on the Ethereum Sepolia public test network suggests that the proposed architecture can be extended beyond experimental environments into practical institutional deployments. Although public blockchain networks introduce additional communication overhead, the observed feasibility of smart contract interaction indicates that the framework remains suitable for real-world credential verification applications when combined with intelligent caching and repository optimization mechanisms.

F. Limitation of the Study

While the evaluation showed measurable improvements in verification responsiveness and blockchain query reduction, several limitations have been identified. The primary scalability evaluation was conducted in a controlled cloud-hosted environment using simulated large-scale

workloads. In addition, the proposed framework was deployed and validated on the Ethereum Sepolia public test network to assess real blockchain communication behaviour, smart contract accessibility, and network latency. However, the study did not evaluate performance under sustained public blockchain congestion, large-scale multi-institution deployments, or prolonged public network fluctuations. Consequently, the reported results should be interpreted as a combination of controlled scalability benchmarking and public testnet feasibility validation.

The workload was a major focus of the evaluation, and the primary attention was paid to the performance behavior at the level of requests, in the context of fresh, mixed, and repeated verification scenarios. The concurrent-user stress testing with many users simultaneously from institutions was not widely tested and could be a topic for future research. The metadata-aware sharding implementation also implemented relatively simple partitioning logic that relies on certain characteristics of the credential metadata. There are still more sophisticated adaptive shard balancing and distributed shard migration techniques that could be explored. The security evaluation was limited to the architectural controls and functional checks available in the prototype. The study did not include independent smart-contract auditing, formal verification, penetration testing, deliberate private-key compromise, RPC manipulation or coordinated denial-of-service testing. The security findings is therefore interpreted as prototype-level validation rather than evidence of production security certification.

VI. CONCLUSION

This study proposed an adaptive optimization framework for blockchain-based academic credential repositories based on a cache reuse mechanism and a repository partitioning mechanism based on metadata. The framework was developed to resolve the scalability and verification responsiveness issues of institutional credential verification systems, especially in the scenarios of repetitive verification requests and increasing repository sizes. Evaluation results showed that the repeated blockchain verification operations could be reduced significantly by adaptive cache-assisted verification under repeated and mixed verification workloads. New verification workloads continued to rely on direct interaction with the blockchain, whereas multiple verification workloads were able to benefit from reduced latency and higher throughput by intelligently reusing previously verified credential responses. Based on these results, it can be concluded that the workload-aware verification optimization has the potential to greatly enhance the responsiveness of academic verification in institutional repositories.

It was also observed from the study that the organization of the repository and the efficiency of the distributed retrieval of the repository improved with metadata-aware partitioning as the repository workload increased. The direct performance benefits of sharding were seen based on

the workload conditions, but the structure of the distributed repository was observed to have better load distribution behavior and minimized dependency on the centralized repository on credential retrieval operations. The proposed framework not only enhances performance but also shows a practical way of how blockchain trust mechanisms can be combined with adaptive distributed systems techniques in the educational environment. The hosted prototype environment also points to institutions being able to improve scalability of verification measurably without having to fully rely on the expensive infrastructure needed to deploy a public blockchain.

DATA AVAILABILITY

The experimental data used in this study consist of synthetic, template-informed academic credential records generated for controlled scalability and performance testing; no real student records or production verification logs were used. The data and supporting benchmark configurations are available from the corresponding author upon reasonable request.

REFERENCES

A blockchain-based digital educational certificate verification system. (2024). *ITEGAM-JETIA, 10*(49), 35–41. https://doi.org/10.5935/jetia.v10i49.1145

Alsobhi, H. A., Alakhtar, R. A., Ubaid, A., Hussain, O. K., & Hussain, F. K. (2023). Blockchain-based micro-credentialing system in higher education institutions: Systematic literature review. *Knowledge-Based Systems, 265*, 110238. https://doi.org/10.1016/j.knosys.2022.110238

Ateniese, G., Magri, B., Metere, R., Spognardi, A., & Venturi, D. (2017). Certificate transparency and decentralized certificate validation using blockchain technology. *Future Generation Computer Systems, 107*, 593–602. https://doi.org/10.1016/j.future.2017.08.024

Chotikakamthorn, N., San, A. M., & Sathitwiriyawong, C. (2024). On-chain verifiable credential with applications in education. *ECTI Transactions on Computer and Information Technology, 18*(3). https://doi.org/10.37936/ecti-cit.2024183.256091

Esguerra, M., Piad, K., Tano, I., Victoriano, J., Espino, J., & Lagman, A. C. (2024). Smart credentialing and verification system for national certificates using blockchain technology. In *Proceedings of the 2024 8th International Conference on Digital Technology in Education* (pp. 183–187). ACM. https://doi.org/10.1145/3696230.3696266

Fernández-Caramés, T. M., & Fraga-Lamas, P. (2019). A review on the use of blockchain for the Internet of Things. *IEEE Access, 6*, 32979–33001. https://doi.org/10.1109/ACCESS.2018.2842685

Gangwar, S., & Chaurasia, A. (2024). Blockchain-based authentication and verification system for academic certificate using QR code and decentralized applications. *International Journal of Computer Applications, 186*(26), 1–10.

Grech, A., & Camilleri, A. F. (2017). *Blockchain in education*. Publications Office of the European Union. https://publications.jrc.ec.europa.eu/repository/handle/JRC108255

Grech, A., Sood, I., & Ariño, L. (2021). Blockchain, self-sovereign identity and digital credentials: Promise versus praxis in education. *Frontiers in Blockchain, 4*, 616779. https://doi.org/10.3389/fbloc.2021.616779

Gupta, S. (2024). On-chain crypto-secured credential verification on permissioned blockchain. In *2024 IEEE International Conference on Blockchain and Distributed Systems Security* (pp. 1–6). IEEE.

Kaneriya, J., & Patel, H. (2023). A secure and privacy-preserving student credential verification system using blockchain technology. *International Journal of Information and Education Technology, 13*(8), 1251–1260.

Khairuddin, I. E., Zaini, M. K., Zakaria, S., & Uzir, N. A. (2024). Credential verification on blockchain: A conceptual framework of Internet of Education for tertiary education. *International Transaction Journal of Engineering, Management, & Applied Sciences & Technologies, 15*(1).

Kleppmann, M. (2023). *Designing Data-Intensive Applications*. O'Reilly Media.

Ma, J., & Fang, Y. (2020). Blockchain-based decentralized credential management system for higher education. *International Journal of Information Management, 52*, 102–118. https://doi.org/10.1016/j.ijinfomgt.2019.102067

Nigeria Data Protection Commission. (2023). *Nigeria Data Protection Act 2023*. Federal Government of Nigeria. https://ndpc.gov.ng

Noshi, T., & Xu, X. (2024). Scalable hybrid blockchain architectures for educational credential management. *Journal of Distributed Systems and Applications, 15*(2), 88–103.

Ocheja, P., Flanagan, B., & Ogata, H. (2018). Connecting decentralized learning records: A blockchain-based learning analytics platform. In *Proceedings of the 8th International Conference on Learning Analytics and Knowledge* (pp. 265–269). ACM. https://doi.org/10.1145/3170358.3170365

Ocheja, P., Flanagan, B., & Ogata, H. (2020). Managing lifelong learning records through blockchain technologies. *Research and Practice in Technology Enhanced Learning, 15*(1), 1–19. https://doi.org/10.1186/s41039-020-00122-9

Okon, E., Umoren, U., & Philips, A. (2023). Digital credential verification challenges in higher education institutions. *African Journal of Information Systems, 15*(3), 45–59.

Redis Labs. (2024). Redis Documentation: Caching Patterns and Performance Optimization. https://redis.io/docs/

Rustemi, A., Dalipi, F., Atanasovski, V., & Risteski, A. (2024). DIAR: A blockchain-based system for generation and verification of academic diplomas. *Discover Applied Sciences, 6*, 297. https://doi.org/10.1007/s42452-024-05984-1

Sharples, M., & Domingue, J. (2020). The blockchain and kudos: A distributed system for educational record management. In D. Ifenthaler & D. Mah (Eds.), *Blockchain in education* (pp. 139–154). Springer. https://doi.org/10.1007/978-3-030-23254-3_8

Verification of education credentials on European Blockchain Services Infrastructure (EBSI): Action research in a cross-border use case between Belgium and Italy. (2023). *Big Data and Cognitive Computing, 7*(2), 79.

Wang, S., Ouyang, L., Yuan, Y., Ni, X., Han, X., & Wang, F. Y. (2020). Blockchain-enabled smart contracts: Architecture, applications, and future trends. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 49*(11), 2266–2277. https://doi.org/10.1109/TSMC.2019.2895123

Yaga, D., Mell, P., Roby, N., & Scarfone, K. (2019). *Blockchain technology overview* (NISTIR 8202). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.IR.8202

Zhao, W., Aldyaflah, I. M., Zheng, Z., & Luo, X. (2024). A blockchain-based academic degree attestation system. *International Journal of Parallel, Emergent and Distributed Systems, 39*(5), 557–571.

Zheng, Z., Xie, S., Dai, H., Chen, X., & Wang, H. (2017). An overview of blockchain technology: Architecture, consensus, and future trends. In *Proceedings of the IEEE International Congress on Big Data* (pp. 557–564). IEEE. https://doi.org/10.1109/BigDataCongress.2017.85

Zimmermann, O. (2016). Microservices tenets: Agile approach to service development and deployment. *Computer Science - Research and Development, 32*(3–4), 301–310. https://doi.org/10.1007/s00450-016-0337-0