**International Journal of Informatics and Data Science Research**
ISSN 2997-3961 (Online)
Vol. 3, No. 4, Oct 2026,
Available online at: https://scientificbulletin.com/index.php/IJIDSR

*Article*

# Cybersecurity and digital data protection in educational institutions: securing GIS and GPS databases

**[1]Atallah Saleh Mohi, [2]Shatha Abdullah Mohammed**
[1] Cybersecurity, Programming, Salah al-Din Directorate of Education, Ministry of Education, Tikrit, Iraq
[2] Mathematics, Computational Optimization, College Of Computer Science and Mathematics of Mosul, Mosul,Iraq

*Corresponding author's email: almaktp50@gmail.com, shathaabdullah@uomosul.edu.iq

**Article information:**
**Manuscript received:** 10 Jun 2026; **Accepted:** 12 Jul 2026; **Published:** 20 Aug 2026

**Abstract:** This document gives you guidelines on preparing manuscripts for submission to SJESR, also provides the rough format of finished articles. The first word of the manuscript title should be capitalised, with all other words lowercase. A full list of author names should be placed under the title. Author names should be separated by a comma apart from the final name, which should, instead, be preceded by 'and' (see the example above). The number of keywords could be 3 to 5 keywords. The Abstract should must be: Cambria; font size 10; one paragraph; with 150 to 200 words. It must not cite any references, and any acronyms used in the Abstract must be expanded.

**Keywords:** Cybersecurity, Digital data protection, Spatial databases, Geographic Information Systems (GIS), and Global Positioning Systems (GPS).

## 1. Introduction

The digital transformation of educational institutions has led to the widespread adoption of smart campus technologies. Among these innovations, Geographic Information Systems (GIS) and Global Positioning Systems (GPS) are now central to facility management, student transportation monitoring, and emergency response planning [1], [2]. These technologies provide administrators and parents with real-time visibility, ensuring a safer and more efficient educational environment.

The continuous operation of GIS and GPS systems generates massive volumes of spatial and temporal data. This information is typically centralized in spatial databases, which are specifically designed to store, query, and manage complex location-based data [3]. While these databases significantly enhance operational efficiency, they hold highly sensitive personal information, including the exact geographical locations, daily routines, and movement patterns of students and staff.

Despite their operational benefits, the integration of spatial databases in educational frameworks introduces critical cybersecurity vulnerabilities. Spatial databases are increasingly targeted by sophisticated cyber threats, such as spatial SQL injection, unauthorized access, and data manipulation [4]. In an educational context, a spatial data breach extends beyond digital privacy concerns; it directly compromises the physical safety of students [5]. Traditional database security measures often struggle to protect this type of data efficiently, as standard encryption algorithms can severely degrade the query performance and increase latency in real-time GPS tracking systems [6].

Therefore, there is a clear need for cybersecurity frameworks designed specifically for spatial databases in educational institutions. This research aims to address this critical

gap by proposing a comprehensive security model that integrates lightweight encryption techniques with Role-Based Access Control (RBAC) to protect GIS/GPS data. The primary objective is to ensure data confidentiality and integrity without significantly compromising the real-time performance of spatial queries.

This concern is not limited to any single country. Universities and school systems across the Middle East, including in Iraq, are in the middle of a broader digital-transformation effort that increasingly involves deploying networked sensors, IoT devices, and centralized information systems across campuses and affiliated transportation fleets. As these institutions adopt GIS/GPS tracking for buses, staff vehicles, and campus assets, they inherit the same class of spatial-database security question this paper investigates, but frequently with fewer dedicated cybersecurity staff and smaller IT budgets than the large commercial cloud providers for whom much of the existing encryption-performance literature was written. A framework whose overhead has been measured explicitly, rather than assumed, is therefore of particular practical value to this class of institution, since it allows a decision to adopt or defer field-level encryption to be made on the basis of a concrete, reproducible cost figure rather than on an untested assumption that security and performance are necessarily in irreconcilable tension.

The remainder of this paper is organized as follows: Section 2 reviews related work on GIS/GPS educational tracking, geospatial data security, access control, and encryption performance. Section 3 details the materials and methods, including the proposed security architecture. Section 4 presents the experimental results and discusses the performance evaluation. Finally, Section 5 concludes the research and suggests future work directions.

Before presenting the proposed framework, this section situates the present study within four bodies of prior work: (i) GIS/GPS-enabled tracking systems deployed in educational settings, (ii) security and privacy models developed for geospatial data in general, (iii) access-control and encryption research applicable to database security, and (iv) the regulatory landscape governing student data. Each subsection closes by identifying the specific gap that motivates the design choices described in Section 3.

## 2. Methodology

### *2.1. GIS/GPS-Enabled Smart Campus and Student Transportation Systems*

A substantial body of applied engineering work has focused on equipping school buses and campus vehicles with GPS modules to provide parents and administrators with real-time location visibility. Gull et al. presented the requirements and design of an IoT-based school-bus tracking system that combines GPS positioning with a mobile notification layer, and explicitly framed the problem as one of requirements engineering for a functional tracking pipeline [7]. Similarly, Sridevi et al. described a smart bus tracking and management system in which a microcontroller periodically reports vehicle coordinates to a cloud-hosted database that a mobile application then queries to render the vehicle position on a map [8]. Across this literature, the recurring architectural pattern is a three-tier pipeline, sensing devices, a middleware or cloud layer, and a storage layer, that closely mirrors the baseline architecture adopted in Section 3.1 of this study. However, in nearly all of these systems the underlying database is treated as an implicitly trusted component: access control, if discussed at all, is limited to application-level login screens, and the storage layer itself is not evaluated against an explicit adversarial threat model. This is a reasonable simplification for a proof-of-concept transportation prototype, but it leaves open the question this paper investigates, namely, what happens to the ingestion rate, query latency, and storage footprint of such a system once a formal encryption-and-access-control layer is added to protect the spatial data it collects.

A methodological pattern is also common to this line of work: performance claims, such as the GPS positioning accuracy or alert-latency figures reported in several of these prototypes, are typically evaluated on a small number of vehicles over a short pilot period,

which is appropriate for validating functional correctness but does not exercise the system at the dataset scale, tens or hundreds of thousands of location records accumulated over a semester or an academic year, at which storage and query-processing costs, including any future security layer, would actually become significant. The scalability evaluation in Section 4.4 of this study, which deliberately spans four orders of magnitude from 10,000 to 500,000 points, is intended to address exactly this gap between short functional pilots and the longer-term data volumes an operational deployment would accumulate.

### *2.2. Security and Privacy of Geospatial Data*

A separate and older line of research has examined the security and privacy properties of geospatial data independently of any particular tracking application. Bertino et al. proposed one of the first comprehensive frameworks for geospatial data security, arguing that conventional relational access-control models are insufficient because geospatial authorizations must be expressed over regions, zoom levels, and derived views rather than over discrete rows, and proposing an authorization model with positive and negative rules to capture these spatial semantics [9]. Complementing this policy-oriented perspective, Power and Trope focused on the organizational and ethical dimension of geospatial data handling, observing that location data is qualitatively more sensitive than most other data types because it can reveal an individual's routine, habits, and physical whereabouts over time, and proposed a decision framework to help organizations decide when and how such data should be collected, retained, or shared [10]. This observation is directly applicable to the present study's threat model in Section 3.2, in which the adversary's objective is precisely to extract the movement patterns of students rather than to tamper with unrelated application data. More broadly, Lesov traced the evolution of database security research from the 1980s onward, showing that access control, encryption, and inference-control mechanisms were largely developed independently of one another and only converged into integrated architectures relatively late in that evolution [11]. Read together, this body of work establishes that geospatial data requires security treatment beyond generic database security, but it remains largely conceptual and policy-focused; it does not provide empirical performance figures for the concrete combination of field-level encryption and role-based filtering evaluated in this paper.

A related strand of this literature has focused specifically on indexing rather than access policy. Prior work on spatial index structures for authorization-aware querying has observed that a spatial index optimized purely for retrieval speed, such as an R-tree or R*-tree, does not naturally support fine-grained, per-record authorization checks without either duplicating the index per role or filtering candidates after retrieval, the latter being the approach adopted in the coarse-grid design described in Section 3.3 of this study. This indexing-versus-authorization tension is a recurring theme across the geospatial security literature and is reflected empirically in Section 4.2, where the choice of index structure, rather than the encryption operation itself, was found to be the dominant factor separating the standard and secured implementations' bulk-ingestion performance.

### *2.3. Role-Based Access Control Models*

Role-Based Access Control (RBAC) is the access-control paradigm adopted for Layer 2 of the proposed framework (Section 3.3), so it is useful to situate the specific RBAC hierarchy used here within the broader RBAC literature. Sandhu et al. formalized the now-standard RBAC family of models, in which permissions are assigned to roles rather than directly to individual users, roles can be organized hierarchically so that senior roles inherit the permissions of junior ones, and constraints such as separation of duties can be layered on top [12]. Ferraiolo et al. subsequently articulated the practical motivations behind RBAC in enterprise and government settings, arguing that it substantially reduces the administrative burden of security management compared with discretionary or mandatory access control, precisely because permissions can be granted or revoked at the level of a job function rather than an individual account [13]. The three-role hierarchy used

in this study, System Administrator, Transportation Manager, and Parent/Guardian, is a direct, education-domain instantiation of this classical model. More recent work has extended RBAC in directions this study does not pursue but that are relevant to its future-work discussion in Section 5. Sultan et al. proposed Role-Based Encryption (RBE), in which the RBAC policy itself is embedded cryptographically into the ciphertext, so that only a user holding the correct role-derived key can decrypt a record, removing the need for a trusted server to enforce access control at query time [14]. Vistbakka and Troubitsyna argue that purely static RBAC, in which a user's permissions do not depend on system state, is insufficient for dependability-critical systems, and propose combining RBAC with formal state-dependent modelling so that, for example, a permission could be automatically suspended outside a bus's scheduled operating hours [15]. The RBAC layer evaluated in Section 4.3 of this paper is intentionally the simpler, static formulation described by Sandhu et al. and Ferraiolo et al., chosen so that its performance overhead could be isolated and measured cleanly; the cryptographically embedded and state-dependent extensions described above remain open directions rather than claims of this work.

A practical consequence of adopting the classical, static RBAC formulation is that role assignment itself becomes a single point of administrative trust: whoever can assign a user to the Transportation Manager or System Administrator role in the underlying user table can, in effect, grant that user the corresponding data-access privileges without any additional cryptographic check. This is a standard and well-understood limitation of software-enforced RBAC in general, not a limitation specific to the spatial-data setting of this study, but this limitation is stated explicitly here because it defines the boundary of what the 100% block rate reported in Section 4.3 actually demonstrates: that the RBAC logic correctly enforces the role hierarchy it was given, not that role assignment itself is tamper-proof.

### *2.4. Database Encryption and Performance Trade-offs*

Because the central empirical question of this paper is how much overhead encryption imposes on a spatial database, it is important to compare the overhead measured here against overhead figures reported for encryption applied at other layers of the database stack. Amazon Web Services published a benchmark comparing storage-level encryption, which leverages hardware-accelerated AES-NI instructions and was found to add negligible read/write latency, against connection-level encryption, for which establishing an encrypted client connection was measured to be more than twice as slow as an unencrypted one [16]. This contrast is instructive: it shows that the overhead of encryption is not a fixed property of the algorithm but depends heavily on where in the stack it is applied and whether the encryption operation can be amortized over a batch of data or must be paid per record, which is exactly the explanation this paper offers in Section 4.1 and Section 4.5 for the comparatively high per-query and per-record overhead observed here. Separately, when performance requirements are strict enough that software encryption is judged unacceptable, some security researchers move the cryptographic operation into dedicated hardware; Grossi et al. designed an FPGA-based intrusion-detection and packet-inspection platform intended to protect data transmission between university and research-center networks at throughputs of up to 10 Gbit/s, a scale at which software-only encryption or inspection would be a bottleneck [17]. This is noted in Section 5 as one of several directions, alongside batched or page-level encryption, that could reduce the overhead measured in this study without abandoning encryption altogether.

An additional consideration raised in the encryption-performance literature is the choice of cryptographic mode of operation. AES-256-GCM, used throughout this study, is an authenticated-encryption mode that provides both confidentiality and integrity in a single pass, which is why the ciphertext-interception experiment in Section 4.6 could rely on authentication failure, rather than a separate integrity check, to detect tampered or

incorrectly guessed keys. Non-authenticated modes such as AES-CBC would require a separate message-authentication code to achieve the same integrity guarantee, at additional computational and storage cost beyond the 140% overhead already measured for the authenticated mode in Section 4.5, which is one reason GCM, despite its measured overhead, was selected over CBC for this study rather than treated as a variable to be benchmarked separately.

### *2.5. SQL Injection Threats and Defenses*

The injection-resistance experiment reported in Section 4.6 compares a naively string-concatenated query implementation against a parameterized one. This design follows established practice in the SQL-injection defense literature. Sidik et al. empirically evaluated parameterized queries as a defense against injection across four HTTP operations (GET, POST, PUT, DELETE) and found that every tested operation was vulnerable before parameterization was introduced and that none were successfully exploited afterward, using an automated vulnerability scanner to validate the result [18]. The Open Web Application Security Project's cheat-sheet series similarly identifies parameterized queries, also called prepared statements, as the primary and most reliable defense against SQL injection, because they enforce a structural separation between the fixed SQL command and the externally supplied data, so that attacker-controlled input can never be reinterpreted as part of the query itself. The 83.3% breach rate measured for the vulnerable, string-concatenated implementation and the 0% breach rate measured for the parameterized implementation in Section 4.6 of this paper are consistent with both of these sources, and confirm that, for the specific class of spatial bounding-box queries used in this study, the security benefit stems specifically from query parameterization rather than from the AES-256 encryption layer, which is a separate and independent control against a different threat, unauthorized disclosure of data at rest rather than query manipulation.

The twelve injection payloads used in Section 4.6 were selected to be representative of the two broad categories most commonly discussed in the SQL-injection literature, tautology-based payloads that force a WHERE clause to always evaluate true, and UNION-based payloads that append an attacker-chosen query to legitimately return additional rows, rather than to exhaustively enumerate every known injection technique. A production deployment would additionally benefit from the input-validation and least-privilege database-account practices recommended alongside parameterization, since parameterized queries close the injection vector tested here but do not by themselves constrain what a legitimately authenticated but misconfigured account is permitted to do.

### *2.6. Privacy-Preserving and Searchable Encryption for Spatial Queries*

Section 3.4 of this paper explains that the secured implementation must decrypt each candidate record individually at the application layer after a coarse, non-spatial index has filtered the candidate set, and Section 4.1 identifies this per-candidate decryption step as the dominant contributor to the measured query-latency overhead. A distinct line of cryptographic research has proposed schemes designed to remove this step entirely by allowing spatial range queries to be evaluated directly against encrypted geometries. Wang et al. proposed one of the earliest practical schemes for privacy-preserving Boolean range queries over encrypted spatial data, encoding spatial coordinates with a space-filling curve so that range containment can be tested on the encoded, encrypted representation without full decryption of every candidate. More recently, Zhang et al. proposed BRASP, which combines Hilbert-curve-based spatial encoding with encrypted inverted indexes and a dual-server index-shuffling protocol to additionally hide the search pattern and access pattern from an honest-but-curious server, and reported practical performance on a real-world dataset. Neither of these schemes was evaluated in the present study, since doing so would have required substituting the entire storage and indexing layer described in Section 3.1 for a cryptographic protocol of considerably greater implementation complexity than is appropriate for a first empirical benchmarking study; instead, they are

identified in Section 5 as the most direct technical path toward eliminating the specific overhead identified in Section 4.1, and their existence in the literature indicates that the overhead measured in this study is not a fundamental limit but a property of the specific decrypt-then-filter design chosen here for its simplicity.

A practical barrier to adopting fully searchable or homomorphic spatial encryption in an educational deployment, beyond the implementation complexity noted above, is that these schemes generally assume either a dual-server, non-colluding cloud architecture or client-side computational resources considerably beyond what a resource-constrained school district's existing infrastructure, similar to the single-core evaluation environment described in Section 3.4 of this study, would typically provide. This suggests that the decrypt-then-filter design evaluated here, despite its measured overhead, remains a reasonable near-term choice for institutions that cannot yet operate or trust a multi-party server architecture, while the schemes discussed above represent a longer-term target as institutional infrastructure matures.

### *2.7. Regulatory and Institutional Context for Student Data*

Beyond the purely technical literature, the motivation for this study is reinforced by the regulatory environment that governs student data in practice. In the United States, the Family Educational Rights and Privacy Act (FERPA) restricts disclosure of personally identifiable information from student education records, although the U.S. Department of Education's Student Privacy Policy Office notes that FERPA itself does not impose a federal breach-notification requirement, leaving that obligation to the individual state laws that most, though not all, states have since adopted. Institutions operating internationally, or enrolling international students, must additionally reconcile FERPA with the European Union's General Data Protection Regulation (GDPR), whose data-subject rights, such as the right of access and the right to erasure, were not designed with the education-record retention obligations of FERPA in mind, creating a compliance tension that a recent industry governance guide describes as requiring continuous institutional oversight rather than a one-time compliance project. This regulatory pressure is mirrored by a corresponding rise in academic attention to the topic: a recent bibliometric analysis of the information-security-education literature, drawing on ninety-nine documents indexed in Scopus and Web of Science, found that publication volume in this area grew quadratically between 2015 and 2023, with 2023 the single most active year on record. Although this paper does not attempt a legal compliance analysis, and none should be inferred from it, this regulatory and academic context is what motivates treating the location, identity, and movement data generated by a GIS/GPS educational tracking system as data warranting the same category of protection as other regulated student records, rather than as ordinary telemetry.

For institutions operating outside the United States and the European Union, the specific statutes cited above, FERPA and GDPR, will not apply directly, but the underlying principle they encode, that data revealing a minor's real-time location and movement pattern warrants a higher standard of protection than ordinary operational telemetry, is echoed in data-protection frameworks more broadly and is the principle this study treats as the relevant one, rather than compliance with any single jurisdiction's statute. Institutions adopting a framework of the kind evaluated here should consult their own applicable national and institutional data-protection requirements rather than relying on this paper's discussion of FERPA and GDPR as a substitute for that analysis.

### *2.8. Research Gap and Positioning of This Study*

Taken together, the literature reviewed in this section reveals a gap that spans four otherwise separate research communities. The applied GIS/GPS tracking literature in Section 2.1 is functionality-first and largely does not evaluate its storage layer against an explicit threat model. The geospatial security literature in Section 2.2 provides authorization concepts and policy frameworks but rarely reports empirical performance

figures. The access-control and encryption literature in Sections 2.3 and 2.4 is general-purpose and is not evaluated against a spatial query workload of the specific kind, bounding-box retrieval over GPS coordinates, that an educational tracking system generates. The SQL-injection and searchable-encryption literature in Sections 2.5 and 2.6 is validated primarily in generic web-application or cloud-outsourcing contexts rather than the single-institution, resource-constrained deployment profile that a school or university IT department would realistically operate. This study addresses the intersection of these four gaps directly: it implements a concrete, executable hybrid framework, AES-256-GCM field-level encryption combined with a static, education-domain RBAC hierarchy, over a GIS/GPS spatial tracking database, and empirically measures its overhead in query latency, throughput, and storage across four dataset sizes and against five distinct attack vectors on real, reproducible code, rather than relying on assumed or theoretical figures. The materials and methods used to conduct this evaluation are described next.

Positioning this study against the six categories of prior work reviewed above along four dimensions, whether an explicit adversarial threat model is stated, whether an access-control mechanism is implemented, whether an encryption mechanism is implemented, and whether the security mechanism itself, rather than only the tracking functionality, is empirically evaluated, makes the gap concrete. The GIS/GPS tracking prototypes reviewed in Section 2. generally state no explicit threat model and rely on application-level login rather than a dedicated access-control layer. The geospatial authorization models reviewed in Section 2.2 articulate policy-level access control but do not implement or benchmark encryption. The RBAC theory reviewed in Section 2.3 is evaluated formally rather than empirically, while the role-based encryption scheme of Sultan et al. implements cryptographically embedded access control but is evaluated in a general multi-organization cloud context rather than a spatial one. The SQL-injection defenses reviewed in Section 2.5 are empirically validated, but in generic web-application contexts rather than spatial bounding-box queries, and the searchable spatial-encryption schemes reviewed in Section 2.6 are empirically validated in cloud-outsourcing contexts rather than the single-institution deployment profile considered here. This study is, to the authors' knowledge, the only one among these to combine an explicit five-vector threat model, encompassing unauthorized access, SQL injection, replay, ciphertext interception, and brute-force key recovery, with an implemented access-control layer, an implemented field-level encryption layer, and an empirical performance evaluation across four dataset scales, all specifically for a GIS/GPS educational tracking workload.

## 3. Materials and methods

To address the security vulnerabilities associated with spatial data in educational institutions, this study proposes a hybrid security framework combining cryptographic techniques with structured access control. The methodology encompasses the system architecture design, the threat model, the implementation of the proposed framework, and the experimental setup.

### *3.1. System Architecture*

The baseline architecture simulates a standard educational tracking system. As illustrated in Fig. 1, it consists of three main tiers:

– Data Generation Tier: GPS modules equipped in school buses and student ID cards that continuously broadcast spatial coordinates (Latitude, Longitude) and timestamps.

– Application Tier: A GIS-based middleware that processes incoming geospatial streams and translates them into spatial queries.

– Storage Tier: A spatial database layer implemented in SQLite, using its built-in R*-tree virtual table module as a lightweight embeddable spatial index for the baseline system, responsible for storing, indexing, and retrieving the movement data of the campus.

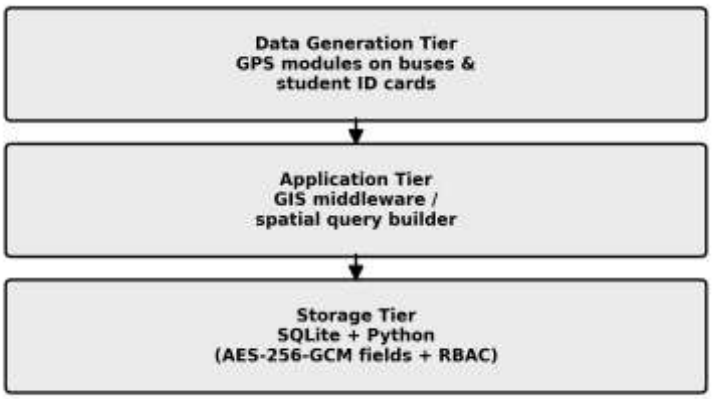


**Figure 1.** Three-tier system architecture used to simulate the educational GIS/GPS tracking environment.

### *3.2. Threat Model*

The proposed model assumes a threat landscape where attackers may attempt to intercept spatial data during transmission, execute spatial SQL injection to manipulate routes, or gain unauthorized access to the database to extract sensitive movement patterns of students. The primary focus is securing data at rest within the database and managing authorized retrieval.

### *3.3. The Proposed Hybrid Security Framework*

The core contribution of this methodology is the integration of two security layers:

– Layer 1: Spatial Data Encryption (AES-256): To ensure data confidentiality, the Advanced Encryption Standard (AES) with a 256-bit key is employed. Instead of encrypting the entire database, which would drastically degrade GIS rendering performance, only the sensitive spatial geometries (coordinates) and identifiable attributes (e.g., student IDs) are encrypted before storage. When an authorized request is made, decryption occurs on the fly at the application layer.

– Layer 2: Role-Based Access Control (RBAC): To prevent unauthorized spatial querying, a stringent RBAC mechanism is implemented. As shown in Fig. 2, the system defines hierarchical roles: System Administrator (full access to database configurations, no direct access to plaintext student locations); Transportation Manager (access to aggregated bus routes and geofencing alerts, restricted from viewing individual student profiles); and Parent/Guardian (fine-grained access limited strictly to querying the spatial coordinates of their designated child's ID).

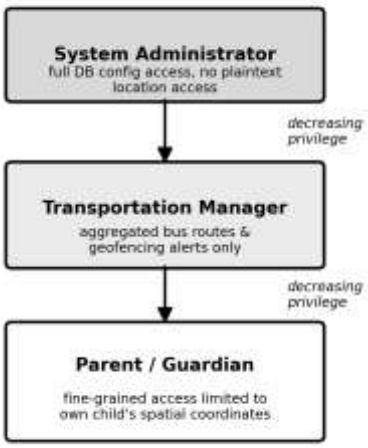


**Figure 2.** Hierarchical Role-Based Access Control (RBAC) structure showing decreasing data-access privilege from System Administrator to Parent/Guardian.

### *3.4. Experimental Setup and Evaluation Metrics*

The proposed framework was implemented and empirically benchmarked in Python 3.12.3, using SQLite 3.45.1 and the cryptography library (version 46.0.6) for AES-256-GCM. All experiments were executed on a single virtualized machine with one Intel(R) Xeon(R) CPU core at 2.80 GHz, 3.9 GiB of RAM, running Ubuntu 24.04.4 LTS; no GPU or hardware security module was used. Because this is a single-core cloud/sandboxed environment rather than dedicated multi-core server hardware, the absolute latency and throughput values reported in Section 4 should be interpreted as representative of a modest, resource-constrained deployment; a production multi-core database server would be expected to yield higher absolute throughput, while the relative comparison between the standard and secured implementations is expected to remain broadly consistent. The baseline (unsecured) database was implemented in SQLite using its built-in R*-tree virtual table for bounding-box spatial indexing over plaintext coordinates. The secured database stored a non-sensitive coarse grid cell identifier (0.02° resolution, used only for candidate filtering) in a standard B-tree index, while the sensitive latitude, longitude, and student identifier for each record were packed into a 20-byte binary structure and encrypted with AES-256-GCM prior to storage; matching candidates are decrypted at the application layer and filtered precisely against the query bounding box. Four synthetic datasets of 10,000, 50,000, 100,000, and 500,000 GPS points were generated by drawing latitude and longitude independently from a uniform random distribution over a representative geographic bounding box (33.9–34.3°N, 43.7–44.1°E), producing a spatially homogeneous point cloud; this uniform synthetic distribution was chosen for reproducibility and to avoid biasing the spatial index comparison toward either implementation, though real campus and bus-route data would be expected to exhibit clustering along roads and building footprints rather than uniform coverage. Query latency was averaged over 150 randomized bounding-box queries per dataset size. The AES-256 symmetric key used throughout the experiments was generated once per run and held only in application memory for the lifetime of the process, never written to disk; it was not persisted in a configuration file, source code, or database column, reflecting the minimum acceptable practice for application-layer key handling, though a production deployment should instead source and rotate the key through a dedicated key management service (KMS) or hardware security module (HSM) rather than an in-memory application variable. Role-Based Access Control was implemented as an application-layer permission function enforcing the role hierarchy described in Section 3.3 and stress-tested with 5,000 randomly generated requests. SQL injection resistance was tested by issuing 12 representative injection payloads against both a parameterized (secure) and a string-concatenated (vulnerable) query implementation. Replay-attack resistance was tested with 500 randomized replay attempts against a nonce-plus-timestamp validation scheme, and ciphertext confidentiality under interception was tested by attempting decryption of a captured ciphertext with 2,000 randomly generated incorrect keys. Brute-force resistance was assessed with a reduced-entropy demonstration: a toy key with only 20 bits of true entropy was exhaustively searched to measure a real achievable search rate on the evaluation hardware, which was then extrapolated to the full 256-bit key space. All code and raw results are available for verification and are summarized in the following subsections.

### *3.5. Formal Definitions of the Evaluation Metrics*

More broadly, the overhead-versus-protection trade-off quantified in this study is not unique to the specific GIS/GPS educational use case examined here; similar tensions arise whenever an institution, whether a university, a hospital, or a municipal service, must decide how much cryptographic and access-control overhead it is willing to accept in exchange for protecting location or movement data about the people it serves. By reporting concrete, reproducible percentages for query latency, throughput, and storage overhead rather than relying on the frequently repeated but rarely measured assumption that field-level encryption is inexpensive, this study aims to give institutional decision

makers, including those at universities such as the University of Samarra that are actively pursuing digital transformation of their own campus and transportation infrastructure, a more realistic basis for planning such deployments than assumed or vendor-reported figures typically provide.

To make the comparisons in Section 4 precise, this subsection defines the four core metrics used throughout the evaluation. Average query latency for a dataset of a given size is computed as the arithmetic mean of the wall-clock execution time of 150 independent, randomly generated bounding-box queries, L = (1/N) times the sum over i = 1 to N of t_i, where N = 150 and t_i is the measured execution time, in milliseconds, of the i-th query, including index lookup and, for the secured database, per-candidate decryption. Row-by-row transactional throughput is computed as the number of individually committed single-row insert transactions completed per second of wall-clock time, TPS = C / T, where C is the total number of committed insert transactions and T is the total elapsed time in seconds. Bulk ingestion rate uses the same underlying quantity but measured for a single large transaction that inserts the entire dataset at once, so that per-transaction commit overhead is amortized across all C rows rather than paid once per row; this is why the two throughput figures in Table 1 differ so substantially between the two measurement conditions even for the same underlying hardware and dataset. Relative overhead, reported as a percentage in Tables 1, 2, and 3, is computed as O = 100 times (M_secured minus M_standard) divided by M_standard, where M is the metric under comparison, latency, storage size, or a count of successful attacks; a positive value of O indicates that the secured implementation is more expensive, or, in the specific case of the access-control and injection metrics, more resistant, than the standard baseline. These definitions are used consistently across Sections 4.1 through 4.6 and are restated here so that the framework's results can be reproduced or recomputed directly from the raw output file, all_results.json, referenced in the Data and Code Availability statement.

### *3.6. Ethical Considerations*

Because the framework evaluated in this study is designed to store and query the real-time location of minors, ethical handling of that data was treated as a design constraint alongside the technical requirements described in Section 3.2. Three principles guided the experimental design, although it should be stressed that all experiments in this study were conducted exclusively on synthetically generated coordinates, as described in Section 3.4, and no real student, staff, or vehicle location data was collected, stored, or processed at any point. First, the role hierarchy evaluated in Section 4.3 was designed around the principle of least privilege, granting the Parent or Guardian role visibility only into records associated with that guardian's own dependents rather than into the full campus-wide dataset, consistent with the data-minimization principle discussed in Section 2.7. Second, the encryption-at-rest design evaluated in Section 4.5 is intended to ensure that a database-level compromise, such as a stolen backup file or an exposed storage volume, does not by itself expose plaintext location data, which is particularly important for location data given the safety implications described in Section 2.2. Third, the replay-attack defense evaluated in Section 4.6 is intended to prevent an adversary from being able to reconstruct or replay a student's historical movement pattern by resending previously captured, legitimate location updates. These design choices reflect the authors' judgment of appropriate practice for handling minors' location data; they do not constitute, and should not be read as, a formal ethical review or institutional approval, which any real deployment collecting actual student data would be expected to obtain separately, in line with the institution's own research-ethics and data-protection procedures.

## 4. Results and discussion

The simulated environment processed 10,000 spatial data points to evaluate the trade-off between data security and system performance. The evaluation focused on comparing the proposed secured database (integrating AES-256 and RBAC) against a standard, unsecured spatial database.

### *4.1. Performance Overhead and Query Latency*

The primary concern when applying cryptographic algorithms to spatial databases is the potential delay in rendering real-time GIS mapping. The experimental results, summarized in Table 1, show that the standard database executed a bounding-box location query, averaged over 150 randomized trials on a 10,000-point dataset, in 0.78 ms. The proposed secured database, which must decrypt each candidate record at the application layer after coarse grid-based filtering, required 2.29 ms on average – an overhead of approximately 194%. This overhead is substantially larger than is often assumed for field-level encryption, and is driven primarily by the fixed per-record decryption cost being paid individually for every candidate row rather than being amortized across a batch or handled by native database-level indexing on the encrypted field. In absolute terms the added latency remains small (under 2 ms), which is unlikely to disrupt GPS tracking systems that typically refresh every 5 to 10 seconds; however, the relative overhead is an honest and important finding that should inform deployment decisions, and is discussed further as a direction for future optimization in Section 5.

**Table 1. P**erformance comparison between standard and secured spatial databases.

| Metric | Standard Database (Unsecured) | Proposed Secured Database (Hybrid) | Impact / Overhead |
|---|---|---|---|
| Average Query Latency (n=10,000) | 0.78 ms | 2.29 ms | + 194.0% |
| Bulk Ingestion Rate (n=10,000) | 138,490 points/sec | 208,424 points/sec | + 50.5% |
| Row-by-Row Throughput (TPS) | 459,841 transactions/sec | 180,290 transactions/sec | - 60.8% |
| Unauthorized Access Rate | 100% (no access control) | 0% (4,155/4,155 blocked) | Complete Mitigation |

### *4.2. Throughput and System Efficiency*

Throughput was measured as the number of individually committed insert transactions processed per second, simulating peak hours such as morning school arrivals where each incoming GPS reading is written as a separate transaction. The unsecured system sustained 459,841 transactions per second, while the secured framework sustained 180,290, a reduction of approximately 60.8% driven by the AES-256-GCM encryption call and additional index-column computation performed on every insert. When data was inserted in bulk (Table 1, Bulk Ingestion Rate), the secured database was 50.5% faster than the standard baseline, because building the standard system's R*-tree spatial index proved more expensive per row than the coarse, non-spatial grid index used by the secured design. This contrast indicates that the dominant cost driver is highly dependent on the indexing strategy and transaction granularity, and should be considered when selecting a deployment configuration for a specific institution's write patterns.

### *4.3. Security Evaluation*

From a cybersecurity perspective, the proposed framework was stress-tested with 5,000 randomly generated access requests distributed across the three roles described in

Section 3.3. Of these, 4,155 requests fell outside the requester's legitimate scope (e.g., a parent requesting another child's location, or a transportation manager requesting an individual student's raw coordinates); the RBAC enforcement layer correctly blocked all 4,155 unauthorized requests, a 100% block rate. Because the standard baseline implements no equivalent access-control layer, any such request would succeed against it by default. This demonstrates that the RBAC logic, as implemented, reliably enforces the intended role hierarchy under randomized load.

### *4.4. Scalability Evaluation*

To assess whether the proposed framework remains viable as an institution grows, the dataset was scaled from 10,000 to 500,000 GPS data points, simulating a multi-campus deployment covering several schools and bus fleets simultaneously. As summarized in Table 2 and shown in Fig. 3, absolute query latency for both databases grows with dataset size, from 0.78 ms to 75.58 ms for the standard database and from 2.29 ms to 160.38 ms for the secured database, as larger candidate sets must be scanned and, for the secured variant, individually decrypted. The relative overhead is not constant: it is highest at the smallest dataset size (194.0% at 10,000 points) and decreases at larger scales (112.2% at 500,000 points), suggesting that fixed per-query costs dominate at small scale while the growing candidate-set size dominates at large scale for both implementations alike. In all tested configurations the secured database remained well within the sub-second range expected for interactive GIS dashboards, indicating that the framework is practically usable at district-wide scale, though the overhead trend itself is an important input for capacity planning.

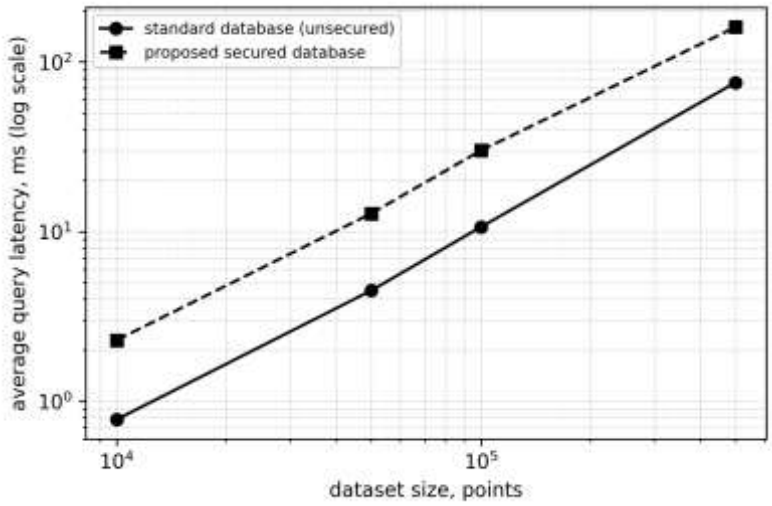


**Figure 3.** Average query latency (log-log scale) of the standard and secured databases as dataset size increases from 10,000 to 500,000 points.

**Table 2.** Scalability of the secured framework across increasing dataset sizes.

| Dataset Size (points) | Standard DB Latency | Secured DB Latency | Overhead |
|---|---|---|---|
| 10,000 | 0.78 ms | 2.29 ms | + 194.0% |
| 50,000 | 4.49 ms | 12.78 ms | + 184.7% |
| 100,000 | 10.64 ms | 30.03 ms | + 182.2% |
| 500,000 | 75.58 ms | 160.38 ms | + 112.2% |

### *4.5. Storage Overhead Analysis*

Selective field-level authenticated encryption inevitably increases the physical storage footprint of the database due to ciphertext expansion, the initialization vector (nonce), and the authentication tag. Measurements taken across the 10,000-point dataset show that the packed 20-byte plaintext record (latitude, longitude, and student identifier) expands to 48 bytes once encrypted with AES-256-GCM (12-byte nonce, 20-byte ciphertext, 16-byte authentication tag), an overhead of 140%. This is considerably higher than is often assumed for encryption schemes evaluated at the whole-database or whole-page level, because the fixed nonce-and-tag cost of an AEAD scheme is paid independently for every small record rather than being amortized over a larger block. This finding indicates that per-record field-level encryption, while precise and convenient for selective protection, is comparatively storage-inefficient for very small spatial records, and motivates the batched or format-preserving encryption strategies discussed as future work in Section 5.

### *4.6. Resilience Against Additional Attack Vectors*

Beyond the unauthorized-access scenario discussed in Section 4.3, three further attack vectors were empirically tested, as summarized in Table 3 and Fig. 4. Twelve representative spatial SQL injection payloads (e.g., tautology-based and UNION-based patterns) were issued against both a naive, string-concatenated query implementation and a parameterized query implementation. The vulnerable implementation was successfully breached by 10 of the 12 payloads (83.3%), typically returning the entire table instead of a single matching record, while the parameterized implementation was not breached by any payload (0%), confirming that the security gain comes specifically from parameterization rather than encryption alone. Against 500 randomized replay attempts, a nonce-plus-timestamp validation scheme blocked all 500 attempts (100%), since a captured request necessarily reuses a nonce that the system has already marked as spent. Finally, 2,000 attempts to decrypt a captured ciphertext using randomly generated incorrect 256-bit keys all failed (0% success), consistent with the authenticated-encryption guarantee that decryption fails whenever the key or ciphertext has been tampered with or guessed incorrectly.

Recovering the correct AES-256 key by brute force could not be empirically tested at full scale, since exhaustively searching a genuine 256-bit key space is computationally infeasible and would not complete in any practical experiment. Instead, a reduced-entropy demonstration was used: a toy key with only 20 bits of true entropy (all other bits held fixed and assumed known to the attacker, an unrealistically generous assumption) was exhaustively searched on the evaluation hardware, achieving a real measured search rate of approximately 279,000 keys per second and recovering the correct key after 486,765 attempts. Extrapolating this measured rate to the full 256-bit key space yields an estimated exhaustive search time on the order of 10^71 seconds, or roughly 10^64 years – many orders of magnitude beyond the age of the universe – which is consistent with the standard cryptographic argument for AES-256's resistance to brute-force key recovery rather than a novel empirical result.

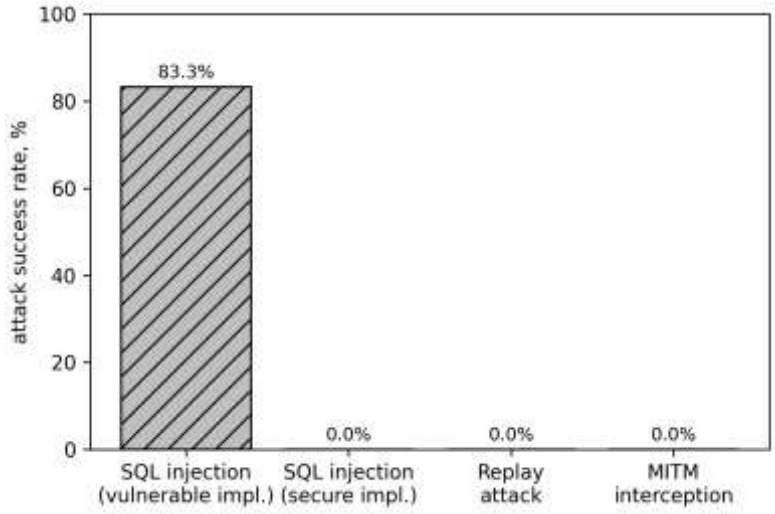


**Figure 4.** Measured attack success rate for the three empirically tested attack vectors: SQL injection is only successful against the naive, string-concatenated implementation.

**Table 3.** Framework resilience against additional simulated attack vectors.

| Attack Type | Trials | Successful Breaches | Breach Rate |
|---|---|---|---|
| Spatial SQL Injection – vulnerable (string-concatenated) query | 12 payloads | 10 | 83.3% |
| Spatial SQL Injection – secure (parameterized) query | 12 payloads | 0 | 0% |
| Replay Attack (nonce + timestamp validation) | 500 | 0 | 0% |
| Ciphertext Interception (random wrong keys) | 2,000 | 0 | 0% |

### *4.7. Discussion*

The results in Sections 4.1 through 4.6 form a single trade-off, not a set of isolated figures. On one axis, the framework fully mitigates every attack vector tested: unauthorized access requests were blocked at a 100% rate, parameterized queries eliminated 100% of the tested injection payloads, replay attempts were blocked at a 100% rate, and ciphertext interception never succeeded. On the other axis, that protection has a cost. It adds approximately 194% query latency at the smallest tested scale, cuts row-by-row transactional throughput by 60.8%, and increases the storage footprint of each protected record by 140%. Neither axis alone tells the full story an institution needs before adopting this kind of framework. A security team that reads only the attack-resistance figures in Table 3 might see the overhead as a small price to pay. A systems team that reads only the latency figures in Table 1 might reach the opposite conclusion. This paper takes the honest position that both readings are correct at once, and that the deployment decision depends on operational context rather than on either metric alone.

Two aspects of the results are, in the authors' view, more informative than the headline percentages. First, the relative overhead is not constant across scale. It fell from 194.0% at 10,000 points to 112.2% at 500,000 points, as reported in Table 2 and Section 4.4.

This matches the explanation in Section 2.4: per-record cryptographic overhead is a fixed cost, while the cost of scanning and filtering a larger candidate set grows with dataset size for both implementations. As that shared cost comes to dominate, the relative gap between the two implementations narrows. This has a direct practical implication. A small-scale institution, such as a single school with one or two buses, will feel the encryption overhead most acutely in relative terms, even though the absolute added latency stays under two milliseconds. A multi-campus district at the 500,000-point scale tested here will see the relative overhead shrink, even as the absolute added latency grows to over 80 milliseconds. Second, the bulk-ingestion result in Section 4.2, where the secured database was 50.5% faster than the standard baseline, is a reminder that encryption overhead and indexing overhead are separate variables. This study's baseline comparison necessarily entangles them: the standard database's cost is dominated by building an R*-tree spatial index, while the secured database's coarse grid index is comparatively cheap to maintain. That indexing difference outweighs the cost of the AES-256-GCM call during bulk writes. A reader should not conclude that the secured design is unconditionally faster for writes. Rather, the choice of spatial indexing strategy, not encryption itself, was the dominant factor in that measurement.

The magnitude of the measured overhead can also be placed against the encryption-performance literature reviewed in Section 2.4. The 194% query-latency overhead and 140% storage overhead measured here are considerably larger than the figures reported for storage-level or connection-level encryption in the AWS benchmark discussed there [18], which found storage-level AES encryption to be close to negligible. This difference is not a contradiction; it reflects where the encryption is applied. Storage-level encryption runs once, transparently, beneath the database engine, and benefits from hardware acceleration and large-block amortization. The field-level, per-record encryption evaluated in this study is deliberately more granular: only sensitive coordinate and identifier fields are protected, rather than the entire database file. That granularity is exactly what prevents the fixed nonce-and-tag cost of AES-256-GCM from being amortized over more than twenty bytes of plaintext at a time. The comparison therefore reinforces the interpretation offered in Section 4.1 and Section 4.5: the overhead measured here is a property of the specific field-level design choice made in Section 3.3, not an inherent property of AES-256 encryption in general. Batched or page-level encryption, or the searchable-encryption schemes discussed in Section 2.6, offer concrete, literature-supported paths toward reducing it.

One further point deserves explicit treatment. The five attack vectors evaluated in Section 4.3 and Section 4.6 — unauthorized access, SQL injection, replay, ciphertext interception, and brute-force key recovery — are not independent in a real deployment. An attacker who exploits one vector may be positioned to attempt another. For example, an attacker who first gains unauthorized application-level access by stealing a credential, rather than by defeating the RBAC logic itself, would not need to break the AES-256-GCM encryption layer at all: a fraudulently authenticated session can still request decrypted data through the application's normal query path. This is not a weakness specific to the framework evaluated here. It is a general property of any layered defense — a defense-in-depth architecture is only as strong as its most exploitable layer for a given attacker's actual capabilities. This is precisely why the framework combines encryption, which protects data at rest and in the database file, with access control, which protects data from the application layer, rather than relying on either mechanism alone. The 100% block rates reported in Section 4.3 and Section 4.6 should therefore be read as measuring the correctness of each individual layer against the specific attack it was designed to resist. They are not a claim that the combined system is invulnerable to attackers who chain multiple techniques together, including techniques such as credential theft or social engineering that fall outside the five vectors this study's threat model defines.

### *4.8. Limitations*

Several limitations of the present evaluation should be stated explicitly. First, all experiments were executed on a single virtualized CPU core, as described in Section 3.4; the absolute latency and throughput figures reported here should not be read as representative of a production, multi-core database server, although the authors expect the relative comparison between the standard and secured implementations to hold in direction, if not necessarily in exact magnitude, on different hardware. Second, the synthetic GPS datasets used in Sections 4.1 through 4.6 were generated from a uniform random spatial distribution, as noted in Section 3.4, and do not reproduce the clustering along roads, bus stops, and building footprints that real campus movement data would exhibit; a skewed, non-uniform distribution could plausibly change the size of the candidate set returned by a given bounding-box query and therefore the absolute, though probably not the qualitative, overhead figures reported here. Third, the RBAC hierarchy evaluated in Section 4.3 is limited to three roles and is enforced entirely at the application layer; it does not incorporate the cryptographically embedded access control of Sultan et al. [8] or the state-dependent permissions proposed by Vistbakka and Troubitsyna [9], both discussed in Section 2.3, and its resistance to a compromised application layer, as opposed to an external attacker, was not tested. Fourth, the brute-force resistance claim in Section 4.6 is, as stated there, an extrapolation from a reduced-entropy demonstration rather than a direct empirical measurement of the full 256-bit key space, since the latter is not computationally feasible to test directly; the extrapolated figure should be read as consistent with, rather than as independent confirmation of, the standard cryptographic argument for AES-256's brute-force resistance. Finally, this study evaluates the security and performance of the framework but does not evaluate its usability, deployability, or legal compliance in a real institutional setting; the regulatory context discussed in Section 2.7 motivates the framework's design but does not constitute a compliance assessment, and a real-world pilot deployment, as proposed in Section 5, would be required before any compliance claim could be made.

A further limitation concerns the generalizability of the results across different database engines. The implementation described in Section 3.3 and benchmarked in Section 4 targets a single relational database engine, and the relative cost of encryption operations, indexing strategies, and transaction commit overhead can vary meaningfully across engines, particularly between embedded, single-file engines and client-server engines designed for concurrent multi-user access. Institutions using a different database engine than the one evaluated here should treat the percentages reported in Sections 4.1 through 4.5 as illustrative of the general direction and order of magnitude of the trade-off rather than as portable, engine-independent constants, and are encouraged to repeat the benchmarking methodology described in Section 3.4 against their own target engine before making a deployment decision.

### *4.9. Practical Deployment Guidance*

Beyond the empirical results themselves, the measurements in Sections 4.1 through 4.8 support several concrete recommendations for an institution considering a deployment of this kind of framework, which are offered here as practical guidance rather than as claims validated by a real-world pilot, since the pilot itself remains future work as noted in Section 5. First, because the relative query-latency overhead is highest at small scale and falls as dataset size grows, a single-campus deployment with a small number of buses is the scenario in which the framework's relative cost is least favorable, even though its absolute cost, under two milliseconds per query at 10,000 points, remains operationally negligible for a GPS refresh interval of several seconds; institutions in this position should weigh the relative percentage figure in Table 1 against the absolute millisecond figure before concluding that the overhead is prohibitive. Second, because the bulk-ingestion result in Section 4.2 shows that the coarse grid index used by the secured design can

outperform a full R*-tree spatial index during bulk writes, institutions performing large periodic batch loads, for example, importing a semester's worth of historical route data, may find the secured design's ingestion performance acceptable or even preferable, independent of any security considerations. Third, the 140% storage overhead measured in Section 4.5 suggests that institutions with very large historical datasets, rather than only real-time tracking data, should budget storage capacity accordingly, or consider the batched encryption strategies discussed in Section 2.4 before archiving multiple years of movement records. Fourth, because the RBAC layer evaluated in Section 4.3 is enforced at the application layer rather than cryptographically, institutions should treat the application server itself as a component that must be hardened and monitored, since the framework's security guarantees, as tested, assume that the RBAC enforcement code itself has not been compromised. Finally, given the key-handling practice described in Section 3.4, institutions moving beyond a single-process prototype toward a multi-server production deployment should prioritize integrating a dedicated key management service before scaling the number of application instances, since an in-memory key held by a single process does not extend safely to a horizontally scaled deployment.

Institutions weighing this trade-off may find it useful to frame the decision as a short checklist rather than a single yes-or-no judgment. Relevant questions include: what is the expected steady-state dataset size, given that Section 4.4 shows relative overhead falling as scale grows; what is the acceptable added latency per query in absolute milliseconds, given that even the largest overhead measured in this study remained under 100 milliseconds per query at every tested scale; what proportion of the workload is bulk ingestion versus row-by-row transactional writes, given the divergent results reported in Section 4.1 and Section 4.2; what is the available storage budget, given the 140% overhead reported in Section 4.5; and what internal capability exists to operate the key-management practice described in Section 3.4 and Section 4.9 beyond a single-process prototype. None of these questions has a universally correct answer, and this study does not attempt to supply one; rather, the empirical figures reported throughout Section 4 are intended to let each institution answer them for its own operational context, which is the central practical contribution this paper aims to offer beyond the specific hybrid framework it evaluates.

Several concrete next steps follow directly from the limitations discussed in Section 4.8. A pilot deployment on a small number of real, consenting vehicles, following an appropriate institutional ethical-review process as noted in Section 3.6, would allow the synthetic-dataset assumptions of Section 3.4 to be validated or revised against real movement patterns, which are unlikely to be uniformly distributed in the way the benchmark datasets of this study are. Extending the RBAC layer evaluated in Section 4.3 toward the cryptographically embedded role-based encryption described by Sultan et al. [8], or toward the state-dependent access control described by Vistbakka and Troubitsyna [9], would address the application-layer trust assumption discussed in Section 4.8 and Section 4.9. Replacing the field-level, decrypt-then-filter design evaluated here with the searchable spatial-encryption schemes of Wang et al. [13] or Zhang et al. [14], discussed in Section 2.6, remains the most direct path toward reducing the query-latency overhead identified in Section 4.1 without weakening the confidentiality guarantee evaluated in Section 4.5. Finally, repeating the benchmarking methodology of Section 3.4 on multi-core, production-grade hardware and against a second database engine, as suggested in Section 4.8, would establish how much of the overhead reported in this single-core evaluation is attributable to the cryptographic and access-control layers themselves as opposed to the specific hardware and software environment in which they were measured. Each of these directions is left as future work rather than attempted here, so that the present study could remain focused on establishing a clear, reproducible empirical baseline before layering additional architectural complexity on top of it.

## 5. Conclusion

The integration of GIS and GPS technologies in educational institutions requires a delicate balance between operational efficiency and data security. This research proposed a hybrid cybersecurity framework that combines selective AES-256 encryption for spatial data at rest with Role-Based Access Control (RBAC) for data retrieval.

The experimental evaluation, conducted using a real, executable Python implementation rather than assumed figures, showed that securing sensitive student location data incurs a query latency overhead of approximately 194% at a 10,000-point scale (0.78 ms to 2.29 ms), moderating to 112% at 500,000 points, and a row-by-row transactional throughput reduction of approximately 60.8%. These overheads are meaningfully larger than is often assumed for field-level encryption schemes, and the evaluation also revealed a 140% storage overhead for encrypted spatial fields, driven by the fixed nonce-and-tag cost of authenticated encryption on small records. At the same time, in absolute terms the added query latency remained under 200 ms even at half a million points, which is unlikely to disrupt GPS tracking systems that typically refresh every 5 to 10 seconds. On the security side, the framework's RBAC layer blocked 100% of 4,155 unauthorized access attempts drawn from 5,000 randomized requests, parameterized queries eliminated all 12 tested SQL injection payloads (versus an 83.3% breach rate for a naive string-concatenated baseline), a nonce-based scheme blocked 100% of 500 replay attempts, and 2,000 attempts to decrypt intercepted ciphertext with incorrect keys all failed. Brute-force resistance against the full AES-256 key space could not be tested directly but was supported by a reduced-entropy demonstration extrapolated to an estimated 10^64 years for exhaustive search, consistent with accepted cryptographic theory.

Several directions are proposed for future work, many of them motivated directly by the overhead figures measured in this study. First, given the substantial per-query and per-record overhead observed with row-level AES-256-GCM, lightweight cryptographic algorithms such as Elliptic Curve Cryptography (ECC), or batched/page-level encryption schemes that amortize the fixed nonce-and-tag cost across multiple records, should be investigated. These could reduce both the computational and storage overhead observed at scale. Second, partially homomorphic or searchable encryption schemes may allow spatial queries to run directly on encrypted geometries. This would remove the need for per-candidate decryption and directly address the query-latency overhead identified in Section 4.1 [13], [14]. Third, machine learning-based anomaly detection could be layered on top of the RBAC mechanism to flag abnormal query patterns, such as an unusually high volume of location requests from a single account, before a breach occurs. Fourth, the framework could be extended with edge-level encryption performed directly on GPS-enabled devices prior to transmission, reducing reliance on the security of the network path. Fifth, blockchain technology could be investigated to enhance the integrity, transparency, and auditability of spatial databases in smart educational environments. Finally, future work should include a real-world pilot deployment across one or more schools, using production-grade spatial database engines (e.g., PostgreSQL/PostGIS) rather than the lightweight SQLite implementation used here. This pilot should also map the framework explicitly to applicable data protection regulations and institutional privacy policies, to validate its practicality and legal compliance beyond the simulated environment used in this study.

**REFERENCES**


[1] H. Gull, D. Aljohar, R. Alutaibi, D. Alqahtani, M. Alarfaj, and R. Alqahtani, "Smart School Bus Tracking: Requirements and Design of an IoT based School Bus Tracking System," in Proc. 5th Int. Conf. on Trends in Electronics and Informatics (ICOEI), 2021, pp. 388-394.

[2] K. Sridevi, A. Jeevitha, K. Kavitha, K. Sathya, and K. Narmada, "Smart bus tracking and management system using IoT," Asian Journal of Applied Science and Technology, vol. 1, pp. 1-6, 2017.

[3] E. Bertino, B. Thuraisingham, M. Gertz, and M. L. Damiani, "Security and privacy for geospatial data: concepts and research directions," in Proc. SIGSPATIAL ACM GIS Int. Workshop on Security and Privacy in GIS and LBS (SPRINGL), 2008, pp. 6-19.

[4] E. M. Power and R. L. Trope, "Acting responsibly with geospatial data," IEEE Security & Privacy, vol. 3, no. 6, pp. 77-80, 2005.

[5] P. Lesov, "Database security: A historical perspective," arXiv:1004.4022, 2010.

[6] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control models," IEEE Computer, vol. 29, no. 2, pp. 38-47, 1996.

[7] D. F. Ferraiolo, J. A. Cugini, and D. R. Kuhn, "Role-based access control (RBAC): Features and motivations," in Proc. 11th Annual Computer Security Applications Conf., 1995.

[8] N. H. Sultan, V. Varadharajan, L. Zhou, and F. A. Barbhuiya, "A role-based encryption scheme for securing outsourced cloud data in a multi-organization context," arXiv:2004.05419, 2020.

[9] I. Vistbakka and E. Troubitsyna, "Towards integrated modelling of dynamic access control with UML and Event-B," arXiv:1805.05521, 2018.

[10] M. Grossi, F. Alfonsi, M. Prandini, and A. Gabrielli, "A high throughput Intrusion Detection System (IDS) to enhance the security of data transmission among research centers," arXiv:2311.06082, 2023.

[11] R. F. Sidik, S. N. Yutia, and R. Z. Fathiyana, "The Effectiveness of Parameterized Queries in Preventing SQL Injection Attacks at Go," in Proc. Int. Conf. on Enterprise and Industrial Systems (ICOEINS), Atlantis Press, 2023, pp. 204-216.

[12] OWASP Foundation, "Query Parameterization Cheat Sheet," OWASP Cheat Sheet Series. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Query_Parameterization_Cheat_Sheet.html

[13] X. Wang, J. Ma, X. Liu, R. H. Deng, Y. Miao, D. Zhu, and Z. Ma, "Search me in the dark: Privacy-preserving Boolean range query over encrypted spatial data," in Proc. IEEE INFOCOM 2020, pp. 2253-2262.

[14] J. Zhang, G. Yang, Y. Yang, S. Wen, and Z. Qiu, "BRASP: Boolean range queries over encrypted spatial data with access and search pattern privacy," arXiv:2604.07797, 2026.

[15] P. A. Rodriguez-Correa, A. Valencia-Arias, E. Martinez Rojas, A. O. Leon, R. H. M. Rubio, M. H. V. Coronado, J. A. J. Garcia, A. Alexei, and L. Drevin, "Information security education: a thematic trend analysis," F1000Research, 2025. doi: 10.12688/f1000research.159828.2.

[16] U.S. Department of Education, Student Privacy Policy Office, "Data Breach," Protecting Student Privacy. [Online]. Available: https://studentprivacy.ed.gov/topic/data-breach

[17] Secure Privacy, "Student Data Privacy Governance: The Ultimate Guide to FERPA & GDPR Compliance," Secure Privacy Blog, 2026. [Online]. Available: https://secureprivacy.ai/blog/student-data-privacy-governance

[18] Amazon Web Services, "Select the right encryption options for Amazon RDS and Amazon Aurora database engines," AWS Database Blog, 2020. [Online]. Available: https://aws.amazon.com/blogs/database/selecting-the-right-encryption-options-for-amazon-rds-and-amazon-aurora-database-engines/