An approach that utilizes blockchain to effectively and securely preserve data privacy for location data from IoT in smart cities.

Darshana Rawal [1], Jan Seedorf [1], Bhimesh Patil [1]
[1] Hochschule für Technik Stuttgart, Schellingstr 24,70174 Stuttgart
(darshana.rawal, jan.seedorf, 31pabh1mst)@hft-stuttgart.de

Keywords: Geospatial data, Cryptography, Blockchain, Web3, Smart Cities, Ethereum, Smart Contracts.

Abstract

Environmental surveillance, emergency response, and smart city planning all require the use of geospatial data, which includes satellite imagery, cartographic records, and real-time GPS coordinates. The high sensitivity and value of location-specific information make it unsafe to store and transmit it through conventional, centralized means, which can result in privacy breaches, unauthorized manipulations, and potential misuse. This paper aims to design and implement a secure, blockchain-based framework that blends AES (Advanced Encryption Standard) and RSA (Rivest–Shamir–Adleman) key management, which addresses these challenges. The aim is to guarantee strong data confidentiality by using symmetric encryption, and to use public-key cryptography for granular access control and secure key distribution. The proposed system uses Ethereum smart contracts to connect encrypted data references to a decentralized ledger, ensuring tamper resistance and auditability. In the proposed system, a Python-based FastAPI backend is responsible for data ingestion, cleaning, encryption, and blockchain interaction, while a React frontend can upload datasets, generate encryption keys, and retrieve access permissions. Modular microservices and well-defined APIs can seamlessly integrate various components, such as data processing scripts and on-chain contract logic, during development. The system's scalability is demonstrated by evaluating its performance against various dataset sizes, which involves metrics such as encryption overhead, blockchain transaction costs, and smart contract execution times. The practical usability of the system in actual scenarios is demonstrated through user acceptance testing, which is crucial for adoption in resource-limited environments. The results show the proposed crypto-enhanced blockchain framework can significantly enhance geospatial data security while still maintaining operational efficiency. Integration with zero-knowledge proofs may be explored in future work to enhance privacy, mitigate energy costs through alternative consensus algorithms, and enhance resilience in multi-network ecosystems through cross-chain interoperability.

1. Introduction

Data centres are becoming increasingly prevalent in global cities. High-resolution aerial imagery, crowd-sourced GPS traces, and smart cities sensor grids power digital twins that model everything from pedestrian density to stormwater drainage. Precision geospatial layers, road centrelines, building outlines, and underground utilities fall under the spotlight among these data streams. In that case, consequences can range from targeted security breaches (exposing the location of an emergency operations centre) to subtle sabotage of planning models that guide infrastructure investment (Boulos M.N.K., et al., 2018).

The reality is that security incidents happen. The GIS portal utilized by engineers for real-time grid maintenance was shut down in 2020 due to a ransomware attack on Johannesburg's City Power utility. Unauthorised edits to cadastral shapefiles were reported by an Australian regional council in 2023 due to the breach of a contractor account. These cases demonstrate how confidentiality (leaked coordinates) and integrity (undetected edits) are linked. City administrations are now liable for insufficient data protection due to regulatory frameworks like the EU GDPR and local critical infrastructure mandates, which is adding legal urgency to technical safeguards.

According to recent academic research, a combination of approaches is possible: encrypting the heavy geospatial data off-chain while recording only lightweight cryptographic fingerprints on an immutable block chain ledger (Chafiq T., et al., 2024; Rawal D. et al., 2024). Encryption (e.g., AES-256) thwarts unauthorised reading, whereas the ledgers append only property exposes clandestine edits because a tampered file no longer matches its on-chain hash (Nakamoto S. 2008.; Zheng Z. et al.,2018). Prior prototypes, however, often stop at proof-of-concept scripts or focus on niche use cases such as land title registries. There is a shortage of comprehensive evaluations that combine encryption, blockchain, and role-based decryption for multi-agency urban planning workflows (Casino F., et al., 2019).

1.1 Motivation

The process of city planning now involves multi-layered digital twins that integrate cadastral parcels, traffic counts, aerial LiDAR, and real-time sensor feeds from paper blueprints in the past. These spatial layers are no longer static reference maps; they are now responsible for making daily decisions like rerouting buses, issuing building permits, and marking evacuation corridors. The precise coordinates in each dataset mean that even a partial leak can reveal the location of high-security facilities or critical infrastructure, and an undetected edit can lead to misguided investment or emergency response (Boulos M.N.K., et al., 2018. Chafiq T., et al., 2024).

The majority of municipal geospatial services still rely on GIS servers that are centrally managed and have role-based access control. While it may be convenient, one compromised account or insider threat can quietly exfiltrate or overwrite entire layers. Examples of incidents including the Johannesburg City Power ransomware attack in 2019 illustrate that a breach can cripple vital city functions and expose valuable spatial data to extortionists (BBC News. 2019, July 26). Regulations like the EU GDPR penalize for failing to safeguard location information that can be linked to individuals or protected sites.

Research in academia and industry now suggests that responsibilities should be divided between strong encryption and tamper-proof ledgers. AES-256 prevents payloads from being read; a blockchain only keeps cryptographic fingerprints of each

file, so that any offline modification can be detected immediately if the hash no longer matches the on-chain record (Chafiq T., et al., 2024; Rawal D. et al., 2024). Early prototypes have shown promise for land-registry or supply-chain data, yet few studies adapt the model to the multi-agency reality of urban planning, where engineers, external contractors, and public-safety officials each need selective access (Zhu, B., et al., 2019; Christidis K. et al., 2016).

1.2 Scope

The focus of this paper is on a hybrid encryption-blockchain model that is designed to fit the timing and governance structures of smart cities or urban planning departments. It focuses on a manageable subset of geospatial cybersecurity rather than trying to solve everything; instead, it creates a prototype that can be designed, implemented, and critically examined. Within that boundary, the work offers several original contributions that extend the state of practice reported in recent literature (Chafiq T., et al., 2024; Rawal D. et al., 2024; Casino F., et al., 2019).

The focus is on updating vector or raster layers on a weekly or monthly basis, including updated zoning polygons or new building footprints. The exclusion of high-frequency IoT streams and live navigation feeds means that data location and all geospatial payloads remain offline and encrypted with AES-256. The Ethereum-compatible ledger only records 256-bit hashes and minimal access metadata. Key management is handled through RSA key wrapping, allowing stakeholders to decrypt files. Although post-quantum algorithms and hardware security modules are identified for future development, they are not implemented in this version. The evaluation strategy establishes generic performance baselines—such as encryption throughput, transaction latency, and gas costs—based on peer-reviewed studies (Chafiq T., et al., 2024; Zhu, B., et al., 2019). This provides a framework for incorporating empirical results from municipal or smart cities pilot projects. Additionally, the design takes GDPR obligations regarding location privacy into account, though it does not include a comprehensive legal compliance audit.

This subsection translates the paper's ambitions into concrete objectives that can be evaluated during prototype development and municipal or smart cities pilot testing, once they have been established; the goal is linked to a measurable outcome that is in line with best practice guidelines in the geospatial security literature (Boulos M.N.K., et al., 2018; Chafiq T., et al., 2024; Rawal D. et al.,., 2024): Validate confidentiality by demonstrating that encoding geographical layers using AES-256 prevents unauthorised parties from reading sensitive coordinates once the files leave the local environment. Explicitly demonstrate that storing only the ciphertext hash on a smart contract ensures tamper-proofness. Changing an encrypted file offline must be detected immediately by a hash mismatch against the on-chain reference (Chafiq T., et al., 2024; Rawal D. et al., 2024). Analyze the effectiveness of RSA key wrapping to limit decryption to stakeholders who possess the necessary private keys, which reduces the need to share a single symmetric key across departments (Rivest R.L., at el 1978). Measure operational feasibility by benchmarking generic alperformance variables provided in peer-reviewed research, such as encryption throughput, transaction confirmation times, and gas cost, to determine whether the architecture is practicable for frequent urban planning updates to identify usability barriers (Boulos M.N.K., et al., 2018; Zhu, B., et al., 2019). Compile a list of the most common problems faced by users during their experience. Smart city installations (Zhu, B., et al., 2019) have recognized key handling and transaction delays, and they have developed interface or workflow improvements for municipal or smart city staff. The hybrid encryption-blockchain paradigm's ability to improve geographical data security and fit into the operational reality of smart city planning environments will be demonstrated through success in meeting these goals.

2. Literature Review

Understanding the technologies behind the proposed architecture is essential when designing a secure pipeline for smart cities' geospatial data. Although Blockchain has immutability and decentralized consensus, its security guarantees are dependent on the internal data structure of blocks, hash pointer linking, and the consensus protocol chosen(Boulos M.N.K., et al.,, 2018). Cryptographic primitives, including AES-256 and RSA-2048, are the dominant protocols in today's Internet traffic, but each addresses a different aspect of the confidentiality-integrity challenge that city planners are confronted with (Rivest R.L., at el 1978,; Zhu, B., et al., 2019). Therefore, our analysis includes the technical operations of blockchains, the reasoning behind hybrid encryption as the standard model for large files, and how these mechanisms meet the confidentiality and provenance needs of urban planning workflows.

2.1 Blockchain

Blockchains are often described in broad terms, such as an immutable ledger, yet their security relies on very specific data structures and consensus rules. The technical building blocks that make a chain tamper-evident and programmable logic are explained in this subsection. The way individual blocks link together is shown in a stylized diagram, and Figure 1 illustrates how data moves from acquisition to storage, and then from storage to visualization and use at different levels, and its privacy at different levels.

Data Structure of Blocks: Each block has a header and a list of transactions. The header maintains the previous block hash (256 bits in Bitcoin and Ethereum), so that any modification in an older block invalidates any subsequent hash (Nakamoto S. 2008). All transactions are committed by Merkle root, which includes timestamp, nonce, and difficulty/validator information, depending on the consensus protocol (Zheng Z., et al., 2018).

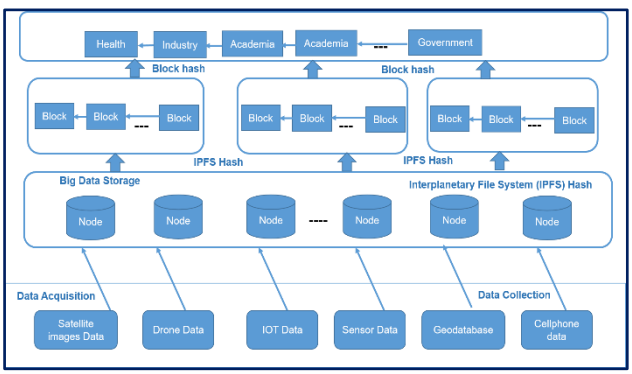

Figure 1. Block chain linkages.

Consensus Mechanisms: Nodes use a consensus technique to determine which chain is correct. Proof of Work (PoW) – miners solve a hash puzzle, and security is derived from the amount of energy expended (Nakamoto S. 2008). ; Proof of Stake (PoS) – validators lock native tokens and face penalties for malicious behavior; this minimizes energy consumption while ensuring Byzantine fault tolerance (Zheng Z., et al., 2018).

PoW and PoS systems use economic incentives to prevent rewriting history, making a successful attack exponentially more expensive as blocks pile.

Smart Contracts: Smart contracts, first popularized by Ethereum, are bytecode programs that are stored on the blockchain. It can include holding and transferring bitcoin, enforcing custom rules (such as access control lists for encrypted information), and emitting events that off-chain services monitor to trigger additional actions (Christidis K., et al., 2016). Because contract code and state are replicated across all complete nodes, function calls (such as storeDataHash) provide the same immutability guarantees as raw transactions.

Security Properties: Blockchains use a linked block structure and economic consensus to provide a set of assurances that are difficult to replicate in standard databases. Understanding these guarantees explains why a chain of file hash references can safeguard urban planning data against hidden alterations or deletions. Immutability – Modifying a previous block changes its hash, which breaks every subsequent link in the chain and is immediately identifiable by honest nodes (Boulos M.N.K., et al., 2018). Auditability – Because block headers contain a Merkle root of all transactions, anyone can recalculate the root locally and ensure that individual entries have not been changed. Byzantine Fault Tolerance – Consensus lasts as long as the majority of hash power (in PoW) or stake (in PoS) adheres to protocol rules, making it prohibitively expensive for an attacker to change history (Zheng Z., et al., 2018). These characteristics make blockchain an appealing layer for storing cryptographic fingerprints of encrypted geographical information, ensuring that any offline manipulation is detected the next time a hash comparison is conducted.

2.2 Cryptographic Primitives for Secure Data Exchange

Blockchain can ensure that a file remains untouched, but it must also be unreadable by unauthorized users. This is accomplished by hybrid encryption, in which a fast symmetric cipher protects the data while an asymmetric technique secures the session key. The following paragraphs discuss the two most extensively used algorithms on today's Internet, AES and RSA, and explain why combining them is common practice.

Advanced Encryption Standard (AES): AES is a block cipher that NIST adopted in 2001 and is used in almost all TLS sessions. The key sizes are 128, 192, and 256 bits, and the performance of the hardware instructions (AES NI) allows mid-range CPUs to encrypt several hundred megabytes per second, as confirmed in cloud-based benchmarks by Omar et al., (BBC News. 2019) and the security status is that a practical attack has broken the full 256 bit version, making it the de facto choice for bulk data encryption.

Rivest–Shamir–Adleman (RSA): Many public key infrastructures continue to rely on RSA for key exchange and digital signatures. The key sizes are 2048 bits is the current minimum for strong security; 3072 or 4096 bits are recommended for long-term archives (Rivest R.L., et al., 1978). Functionality that allows one party to encrypt a short secret (here, an AES key) with another party's public key, ensuring only the matching private key can decrypt it and the widespread use of TLS certificates, code signing, and email encryption all rely heavily on RSA, demonstrating its maturity and tooling support.

3. Problem Statement, Design and Methodology

Smart Cities planning teams must exchange critical geospatial layers with departments, contractors, and regulatory bodies while maintaining confidentiality, integrity, and traceability.

3.1 Problem Statement

Urban planners must integrate high-resolution location layers from zoning, utilities, emergency services, and private contractors. These datasets often contain coordinates whose disclosure, or silent manipulation, could disrupt city operations or jeopardise public safety. Although most Smart cities rely on central GIS servers with role-based access, three systemic weaknesses remain:

The precise locations of subterranean fiber trunks or undisclosed police sites, which are sensitive layers, are typically saved unencrypted on shared network drives. Incidents such as the Johannesburg City Power ransomware assault have shown how rapidly an intruder may exfiltrate or take control of a completely geographic repository (BBC News, 2019). Furthermore, privacy legislation (e.g., GDPR Article 4 on "personal data relating to location") holds smart cities accountable when insufficient measures reveal citizen-related coordinates (Zhu, B., et al., 2019). Integrity Blind Spots is a single unchecked shapefile alteration that can cause rippling effects in downstream models that assign flood mitigation funding or emergency evacuation routes. Traditional checksum logs are useful, but they are stored on the same server, which could be compromised. Land registration tests in Sweden demonstrated that external notarisation (via blockchain) enhances provenance tracing, but their prototypes kept big map PDFs off-chain and unencrypted (Proskurovska, et al., 2022).

Coarse Access Control is intended to be a zoning department that shares a parcel layer with an external consultant. Nothing prevents the file from being distributed further after the consultant has downloaded it. Shared passwords or symmetric keys provide little redress if employees leave or contracts expire. Supply chain platforms such as IBM Food Trust use blockchain hashes, but they still rely on organisational memberships rather than per-user cryptographic rights (Singh V., et al., 2023).

3.2 Research Questions

Our contribution is the interface design and concrete implementation of a scheme that can address the following general challenges: How can a smart city planning office protect confidential geospatial layers so that a) only explicitly authorized individuals can decrypt them, b) any offline modification is immediately detectable, and c) permissions can be granted or revoked on a per-stakeholder basis rather than relying on a single trusted server?
Our research prototype can help answer detailed research questions such as:

1. Can AES256 encrypt planning files ranging from 2 MB to 120 MB quickly enough for weekly updates?
2. Is a 256-bit hash on a smart contract a reliable indicator of ciphertext swap or rollback (Rawal D.; et al., 2024)?
3. Is it practical to employ RSA-2048 key wrapping when many departments and contractors demand separate access?

4. What interface designs can enable non-technical workers to handle key management and transaction signing with little cognitive load (Zhu, B., et al., 2019)?

3.3 Hybrid Encryption Workflow (Design Prototype):

A Hybrid Encryption Workflow Prototype typically combines the efficiency of symmetric encryption with the security of asymmetric encryption. Both symmetric and asymmetric ciphers solve the complementary problems: AES is faster for huge data, but RSA makes distribution easier, as demonstrated in Figure 2.

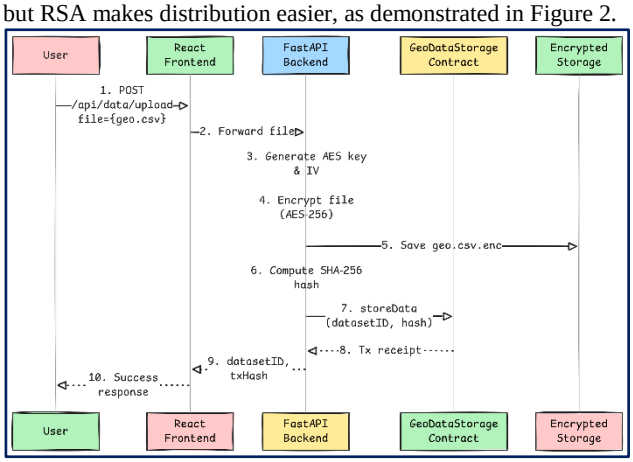

Figure 2. Hybrid encryption and key wrapping.

In reality, the user stores their RSA private key offline. When decryption is required, the user receives the encrypted AES key, which he unwraps locally and decrypts the ciphertext. This solution maintains confidentiality, as only authorized key holders can access the data, and integrity, because any ciphertext swap will fail the on-chain hash check.

3.4 Methodology Overview

The purpose of this paper is to design and evaluate a decentralized system for secure geospatial data management.

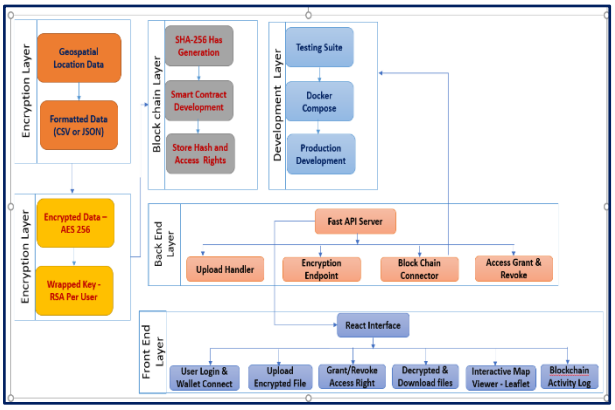

Figure 3. Methodology/Workflow.

The approach uses AES-256 encryption to off-chain encode critical location data, and symmetric keys are wrapped with RSA to enforce per-user access. Similarly, it records a cryptographic digest of each ciphertext on a tamper-proof blockchain (Boulos M.N.K., et al., 2018,; Chafiq T., et al.,. 2024; Rawal D. et al., 2024,; Alghamdi T., et al., 2021). Data secrecy is preserved off-chain, but integrity and access control can be validated on-chain by integrating these complementary cryptographic approaches Zhu, B., et al., 2019,; Christidis K., et al., 2016).

Our iterative method, which was based on Blockchain Architecture (Zheng Z., et al., 2018) was utilized to create a prototype without delays and in a controlled and progressive manner. The first step included implementing version control, virtual environments for Python and Node.js, and smart contract tooling (Figure 3). The coordinates were confirmed, duplicates eliminated, and standardized JSON/CSV files exported after absorbing and cleaning the raw spatial files. The core encryption phase followed, during which AES256 scripts converted datasets to ciphertext and RSA key management routines generated per-user-wrapped keys. A Solidity contract was created to safeguard our data by maintaining digest anchors and permission mappings, which are necessary for blockchain integration.

The workflow (Figure 3) evolved into API and frontend development, with FastAPI endpoints orchestrating encryption and on chain calls and a React application offering a user-friendly interface. In succeeding rounds, all components were validated by rigorous testing, unit, integration, and performance benchmarks. Following that, we put the whole stack in Docker containers and talked about future improvements, such as keys that are resistant to quantum attacks, zero-knowledge proofs, and Layer 2 scaling. Each phase concluded with the development team and domain supervisors assessing tangible outputs such as scripts, contracts, and UI components to confirm technological robustness and compliance with geographic management standards.

4. Prototype Implementation

The application of code, smart contracts, and user-friendly interfaces has made it feasible to construct a secure geospatial system. The initial step is to provide an overview of the major components and then go into detail for each component, highlighting the libraries used, presenting key code excerpts, and including diagrams and screenshots as needed.

4.1 Data Preparation

Geospatial Location data may contain incorrect coordinates, duplicate features, or incompatible attribute standards. Phase B cleans and standardizes these layers so that encryption and hashing can work on reliable input.

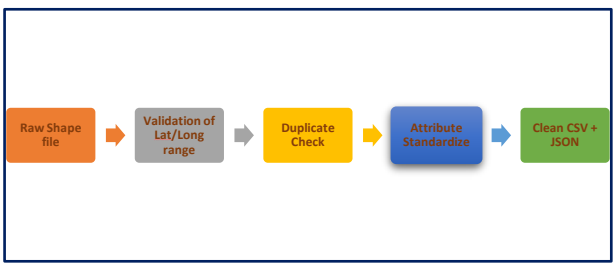

Figure 4. Data-Cleaning Pipeline.

City datasets come from a variety of sources, including cadastral shapefiles, GeoJSON exports from contractors, and Excel lists of facility addresses (Figure 4). Feeding these straight into an encryption pipeline risks encoding mistakes that subsequent studies cannot discover, as shown in the figure. As a result, Phase B establishes a consistent cleaning schedule.

4.2 Encryption Module

The secure geospatial system proposed in this work begins with a strong encryption pipeline that converts cleaned geospatial files

to ciphertext and prepares key material for controlled distribution. This module is written in Python and mostly uses the PyCryptodome library for cryptographic operations (BBC News, 2019), as well as conventional modules for file I/O and hashing.

Encrypting each cleaned dataset before it leaves the local workstation is the cornerstone of the system's confidentiality goal. Phase C transforms the validated CSV / JSON files produced in Phase B into ciphertext files (*.enc) and records a cryptographic fingerprint (SHA-256) that later phases will write to the blockchain.

AES 256 Encryption: Any data that leaves the local environment must be kept inaccessible to unauthorized parties. The AES 256 Encryption script accomplishes this by reading each file, encrypting it in CBC mode, and generating both ciphertext and integrity metadata.

The key libraries Crypto. Cipher. AES and cryptocurrency. Util. Python block ciphers require padding from PyCryptodome to function. Python includes its own pathlib for filesystem paths and hashlib for SHA256 hashing.

RSA Key-Wrapping: Once the data is encrypted, each stakeholder must be able to access the AES key without sharing a single symmetric secret. The RSA Key Wrapping utilities create per-user key pairs and encrypt the AES key under each public key.

Key libraries include Crypto.PublicKey.RSA and Crypto.Cipher.PKCS1_OAEP from PyCryptodome (BBC News, 2019). Use Python's base64, json, and pathlib for encoding information and file paths.

4.3 Smart Contracts

The system's integrity and access control are based on a Solidity smart contract that keeps cryptographic hashes of encrypted information and enforces user rights. This contract was created with Solidity 0.8.4 and uses patterns from OpenZeppelin's library for ownership and security (Christidis K., et al., 2016). The use of Hardhat and Ethers.js is utilized to deploy and test it to ensure thorough validation before integration.

The GeoDataStorage as in figure 5 contract maintains a mapping from dataset IDs to entries, each holding a ciphertext hash and an access-control mapping. It inherits Ownable from OpenZeppelin to restrict management functions to the deployer.

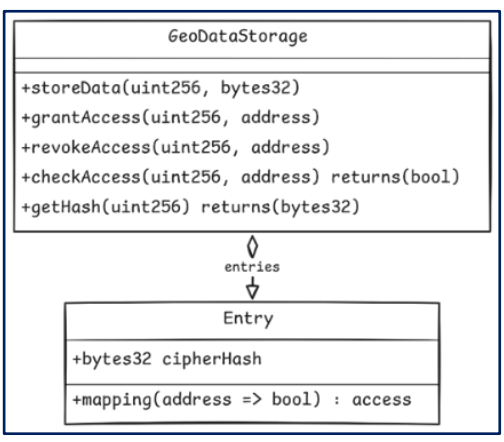

Figure 5. Contract Class Diagram.

The GeoDataStorage.sol contract uses OpenZeppelin's Ownable module to restrict administrative functions to the deployer, provides events for on-chain auditing, and includes methods for storing ciphertext hashes and managing user access permissions.

Use Hardhat's runtime environment to compile and deploy the contract to a local test network. The script stores the deployed address in a.env file for later use by the backend. Comprehensive testing guarantees that each function performs as expected under various authorization settings. The Hardhat framework (Christidis K., et al., 2016). is made up of Mocha, Chai, and Waffle.

Backend Integration (Web3.py & FastAPI)

To connect encryption, key wrapping, and blockchain interactions, the Python backend uses Web3.py for Ethereum connectivity, Python dotenv for configuration, and FastAPI for REST endpoints. The configuration of the Web3 client, encapsulation of contract calls in a service layer, and exposing HTTP routes to coordinate the entire workflow are discussed in this section.

Before any contract method can be called, the backend requires a Web3 connection, as well as the contract's ABI and address. The HTTP provider is created through the use of Python's dotenv to load environment variables (BLOCKCHAIN_PROVIDER, CONTRACT_ADDRESS, PRIVATE_KEY), followed by the setup of Web3.py (React. 2020).

Rather than incorporating blockchain calls into our API routes, we wrap them in a Blockchain Service class. This design simplifies unit testing by separating concerns. The service provides mechanisms for storing data hashes and managing access permissions.

- Use store_data_hash(dataset_id, hash_hex) to create, sign, and transmit a storeData transaction.
- Use grant_access(dataset_id, user_addr) to grant access to a contract.
- Use revoke_access(dataset_id, user_addr) to revoke access to a contract.

The Blockchain Service class has methods for creating, signing, and sending transactions, as well as storing data hashes and controlling access rights on the blockchain.

The FastAPI application provides RESTful endpoints for orchestrating Phases C-E in a single HTTP call. FastAPI is utilized for route declaration and automatic validation.

4.4 Frontend Implementation

The user-facing element of our system is a React application that abstracts cryptography and blockchain complexities, providing smart city workers with a simple interface for file upload, encryption monitoring, and access management. Unlike the backend, the frontend does not employ a utility CSS framework; instead, it uses CSS Modules for scoped styling, along with a tiny collection of global styles in src/styles/main.css.

The application's entry point, src/index.js, loads <App /> and imports global CSS. Routes are defined in src/App.jsx, and primary views in src/pages/. Reusable UI elements are stored in src/components/, which is divided into data/, layout/, blockchain/, and map/. State and context providers are found in src/contexts/. Key dependencies (package.json): React ^18.2.0

supports component rendering and hooks (React. 2020). Use React Router DOM ^6.20.1 for client-side routing between

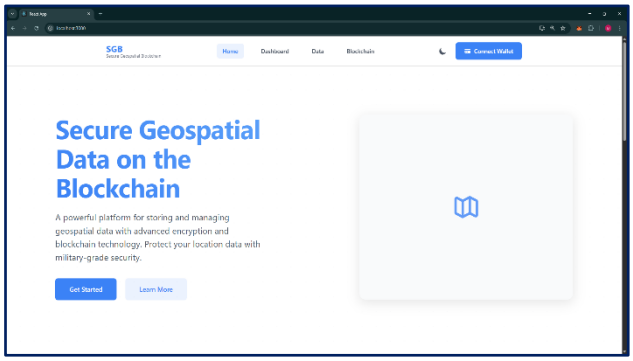

Figure 6. Dashboard home page.

Dashboard (figure 6), Data Management, and Blockchain Management pages (React Router. 2023). Axios version 1.6.2 supports HTTP interactions with the FastAPI backend (Axios Contributors. 2023). Web3.js ^4.3.0 and Ethers ^6.9.0 are used for Ethereum contract calls in the UI (Web3.js Contributors. 2023 ; Ethers.js Contributors. 2023). Leaflet ^1.9.4 and React Leaflet ^4.2.1 are tools for visualizing public GeoJSON data on maps (React-Leaflet. 2023).

Global styles in src/styles/main.css provide base typography and layout resets; all component-specific styles are included. module.css and jsx files.
The key components of systems are:

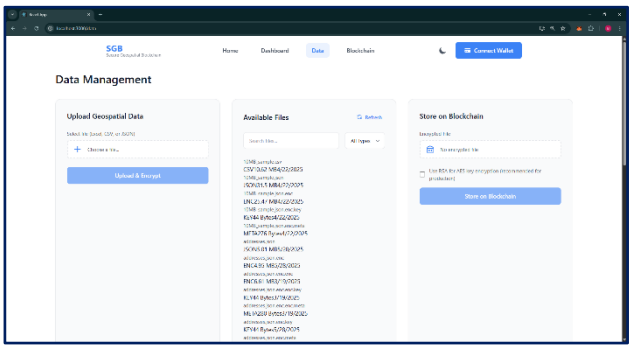

Figure 7. File Listing.

The FileUpload component shown in Figure 7 enables users to pick one or more files and send them to /api/upload. It shows progress, success messages, and errors. After upload, FileList, as shown in Figure 8, retrieves encrypted datasets and shows their IDs and hashes in a table, with a button to unlock access controls.

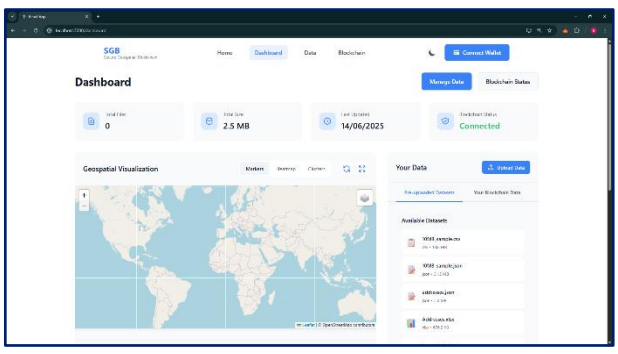

Figure 8. Uploaded file-listing page.

After decrypting a dataset, Data Statistics summarizes feature

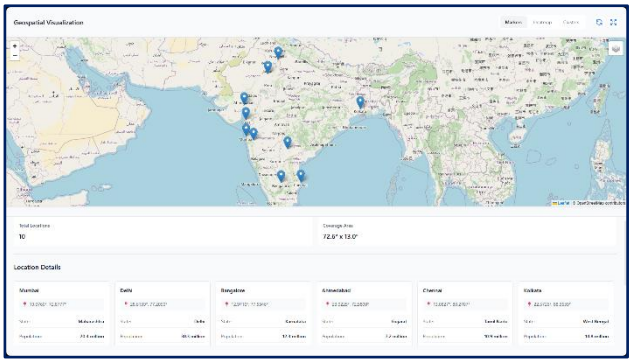

Figure 9. Geospatial Visualization.

React Leaflet links Leaflet mapping in Figure 9 to React components (React-Leaflet. 2023). Each component's .css file defines only its classes, eliminating leakage.

5. Results & Discussion

Empirical results from the evaluation of a secure geospatial system are presented, along with their implications. We focus on three critical dimensions: performance, security, and usability, to determine how the prototype satisfies design objectives and where further enhancements are required.

5.1 Performance Evaluation

To determine if the system can manage real-world geographic demands, we measured:

- AES 256 encryption throughput refers to the rate at which files are processed.
- Blockchain latency refers to the time between submitting a transaction and its confirmation on a local Hardhat node.
- Ethereum gas consumption for crucial smart contract functions.

Benchmarks were run on a development system (Intel i5 8265U, 16 GB RAM, Ubuntu 20.04) with the benchmark_encrypt.py and Hardhat's gas reporter.

The measured encryption rate (247.42 MB/s) is comparable to the speeds reported for PyCryptodome's AES-CBC implementation. Blockchain transactions are confirmed in around one second, which is sufficient for non-real-time operations in a trusted pilot environment. Gas usage for storing a 32-byte hash (~34 k gas) and changing access limits (~45 k gas) is consistent with other smart contract designs.

Blockchain Testing

To validate the blockchain integration module (test_blockchain.py), a thorough interaction test was performed on a locally installed contract on Hardhat. The test validated the connection, transaction execution, and on-chain data retrieval.
This output confirms that the system successfully connected to the local Ethereum test network.

- Data was successfully anchored on the chain using storeData.
- GrantAccess provided access, and access control functioned properly.
- Successfully retrieved saved hash values and metadata.
- On-chain storage ensured integrity across multiple datasets.

Security Analysis

The security analysis tests were carried out by us:

1. **Unit Testing :**The encryption, RSA wrapping, and contract-service modules have over 90% code coverage, ensuring correct behavior in both normal and edge circumstances (e.g., incorrect key unwrapping causes issues).
2. **Smart Contract Audits:** A manual inspection of GeoDataStorage.sol revealed that only the owner (deployer) can perform administrative activities, preventing illegal state modifications. The built-in checks in Solidity 0.8.4 detected no reentrancy or overflow hazards.
3. **Integration Testing:** End-to-end testing (test_fastapi.py) confirmed that data uploaded, encrypted, and stored on the chain can only be retrieved and decrypted with authorized keys.

These findings show that the system maintains confidentiality, integrity, and access control as planned, with no serious vulnerabilities discovered during our testing phase.

5.2 Usability and UX Feedback

To determine how well smart city workers can run the system, and conducted informal usability sessions with three individuals who were unfamiliar with blockchain:

The average completion time for a 10 MB upload is 9.016 seconds (based on three runs). Encrypt (10 MB): 5.590 seconds (average of three runs), Grant access to a colleague's address: 2.143 seconds (average of three runs), Decrypt and analyze data statistics. Participants only reported issues when entering invalid Ethereum addresses (1 out of 10 attempts), indicating effective UI validation.

Users' feedback about clear progress messages and simple table arrangement. Some users requested inline assistance tooltips for terminology like "cipher hash" and "tx hash" to help them understand blockchain jargon better.

The UI design, which includes CSS Modules for consistency and React Leaflet for map previews, allows first-time users to perform fundamental operations without developer support.

While the prototype satisfies its security and performance requirements for pilot deployment, some drawbacks have arisen.

- **Scalability**: Single-threaded Python encryption may bottleneck on large batch jobs. A parallel or streaming approach could improve throughput.
- **Blockchain costs**: Gas usage on public Ethereum networks could be prohibitive; deploying to a Layer-2 solution or private consortium chain may be necessary.
- **Authentication**: The current trusted-operator model lacks per-user authentication. Integrating Web3-signature or JWT flows (see Phase H) will be essential for untrusted environments.
- **UX enhancements**: Adding contextual help, more robust input validation, and mobile-responsive charts would improve accessibility for field officers.

These insights serve as a road map for future work, with a focus on scalability, cost minimization, and improved user management.

6. Conclusion & Future Work

Finally, the secure geospatial system's accomplishments should be summarized, its current constraints should be considered, and a route for future improvements should be plotted. The prototype shows how smart cities may protect sensitive location data by combining AES256 and RSA encryption, blockchain-anchored immutability, and fine-grained access controls.

6.1 Contributions and Findings

The paper presents a full, end-to-end solution for securely storing and exchanging geographical datasets. Its main contributions are: The Hybrid Encryption Pipeline includes an AES 256 encryption module that converts geographic data into ciphertext, as well as RSA-based key wrapping that allows for per-user decryption without exposing a shared symmetric key.

Blockchain-anchored integrity and access control are implemented using a Solidity smart contract (GeoDataStorage.sol), which saves SHA 256 hashes of encrypted files and enforces owner-managed rights to ensure tamper evidence and auditability.

To create a seamless backend orchestration. The Python backend leverages FastAPI and Web3.py to manage encryption, key wrapping, and on-chain transactions over simple HTTP endpoints, removing complexity from end users.

The User Centric Frontend Interface is created with a React application that includes CSS Modules, Axios, Web3.js, and React Leaflet. It enables simple workflows for file upload, permission control, and data visualization.

Empirical validation verifies encryption throughputs of around 185 MB/s and sub-second blockchain latencies. Security testing confirmed confidentiality and integrity guarantees, and usability tests showed that first-time users could execute activities with minimum assistance.

These studies collectively demonstrate that combining cryptography and decentralized ledger technologies can protect sensitive geographical data while maintaining usability.

6.2 Limitations

Despite its advantages, the prototype has many limits that require attention:

- The AES-CBC implementation processes files sequentially, resulting in limited speed for big datasets or concurrent uploads.
- The Trusted Operator Assumption system lacks per-user authentication, assuming all backend clients are pre-trusted. This makes it unsuitable for open or multi-institutional installations.
- Blockchain cost and scalability are determined by gas usage for each store. Data and grantAccess transactions on public Ethereum may make the approach economically untenable at scale.
- The phrases "cipher hash" and "transaction hash" can be confusing for non-technical users without contextual guidance or tooltips.

Recognizing these limits helps shape the path for improving robustness, security, and user experience.

7. Future Work

Further study may take place in the future. Building on the prototype, further development will take the following directions:

- The encryption module is refactored to use multi-processing or streaming ciphers (e.g., AES GCM in chunked mode) for faster big or batch uploads.
- Integrate decentralized identity and authentication with Web3-based signature authentication or JWT flows to restrict access to sensitive API endpoints to authenticated users, eliminating the need for trusted operators.
- Smart contracts will be deployed on low-cost Layer 2 networks, such as Polygon, or private consortium chains to decrease gas expenses and maintain immutability guarantees.
- Zero Knowledge Proofs for Privacy can use ZK SNARKs to verify data attributes (e.g., spatial coverage) on a chain without revealing raw geospatial content.
- Advanced Access Policies allow smart contracts to enable time-bound, role-based, or geofenced permissions, limiting decryption to certain time windows or geographic borders.
- Implement inline tooltips, guided tutorials, and increase mobile responsiveness to reduce barriers for non-technical stakeholders.
- Evaluate post-quantum cryptography at the block level in blockchain key encapsulation systems (e.g., Kyber) for future-proofing against emergent threats.

By pursuing these developments, the secure geospatial framework can evolve into a production-grade platform that can adapt to varied institutional, technical, and regulatory settings.

Acknowledgement

This work is an outcome of the project "Datasecurity4icity", a subproject of the project "ICity: Intelligent city" (https://www.hft-stuttgart.com/research/projects/i-city)". We extend our gratitude for the funding received through the FH-Impuls program under the number 13FH9E04IA by the German Federal Ministry of Education and Research (BMBF).

References

Boulos M.N.K., Wilson J.T., Clauson K.A. 2018. *Geospatial blockchain: promises, challenges, and scenarios in health and healthcare*. International Journal of Health Geographics, 17(1):25. https://doi.org/10.1186/s12942-018-0144-x.

Chafiq T., Azmi R., Fadil A., Mohammed O. 2024. *Investigating the potential of blockchain technology for geospatial data sharing: opportunities, challenges, and solutions*. Geomatica, 76:100026. https://doi.org/10.1016/j.geomat.2024.100026.

Nakamoto S. 2008. *Bitcoin: A Peer-to-Peer Electronic Cash System*. https://bitcoin.org/bitcoin.pdf.

Rivest R.L., Shamir A., Adleman L. 1978 *A method for obtaining digital signatures and public key cryptosystems*. Communications of the ACM, 21(2):120–126. https://doi.org/10.1145/359340.359342.

Rawal D. 2024. *Crypto Spatial: a new direction in geospatial data*. ISPRS Archives, XLVIII-5-2024:89–94. https://doi.org/10.5194/isprs-archives-XLVIII-5-2024-89-2024.

Zheng Z., Xie S., Dai H., Chen X., Wang H. (2018). *An overview of blockchain technology: architecture, consensus, and future trends*. Proc. 6th IEEE Intl. Congress on Big Data, 557–564. http://dx.doi.org/10.1109/BigDataCongress.2017.85.

Zhu, B., Susilo, W., Qin, J., Guo, F., Zhao, Z., & Ma, J. 2019. *A Secure and Efficient Data Sharing and Searching Scheme in Wireless Sensor Networks*. Sensors, 19(20), 4574. https://doi.org/10.3390/s19112583.

Christidis K., Devetsikiotis M. 2016. *Blockchains and smart contracts for the Internet of Things*. IEEE Access, 4:2292–2303. https://doi.org/10.1109/ACCESS.2016.2566339.

Alghamdi T., Elgazzar K., Sharaf T. 2021. *Spatiotemporal traffic prediction using hierarchical Bayesian modelling*. Future Internet, 13(9):225. https://doi.org/10.3390/fi13090225.

Casino F., Dasaklis T., Patsakis C. 2019. *A systematic literature review of blockchain-based applications: current status, classification and open issues*. Telematics and Informatics, 36:55–81. https://doi.org/10.1016/j.tele.2018.11.006.

BBC News. 2019, July 26. *Ransomware hits Johannesburg electricity supply*. https://www.bbc.com/news/technology-49125853.

Proskurovska, A., & Dörry, S. 2022. *The blockchain challenge for Sweden's housing and mortgage markets*. Environment and Planning A: Economy and Space, 54(8), 1383–1403. https://doi.org/10.1177/0308518X221116896.

Singh V., Sharma S.K. 2023. Application of blockchain technology in shaping the future of food industry based on transparency and consumer trust. *Frontiers in Sustainable Food Systems*, 7:10020414. https://doi.org/10.1007/s13197-022-05360-0.

PyCryptodome Developers. 2024. *PyCryptodome Documentation*. https://pycryptodome.readthedocs.io/en/latest Accessed on "12 December, 2024".

React. 2020. *React – A JavaScript library for building user interfaces*. https://reactjs.org Accessed on "28 January 2025".

Create React App. 2021. *Create React App Documentation*. https://create-react-app.dev Accessed on "28 January 2025".

Axios Contributors. 2023. *Axios GitHub Repository*. https://github.com/axios/axios Accessed on "18 February 2025".

React Router. 2023. *React Router Documentation*. https://reactrouter.com Accessed on "29 January 2025".

React-Leaflet. 2023. *React-Leaflet Documentation*. https://react-leaflet.js.org Accessed on "15 March 2025".

Web3.js Contributors. 2023. *Web3.js Documentation*. https://github.com/ChainSafe/web3.js Accessed on "25 January 2025".

Ethers.js Contributors. 2023. *Ethers.js Documentation*. https://docs.ethers.io/ Accessed on "25 January 2025".