Provisional Patent Draft

1. Title of the Invention

System and Method for Mathematical Lineage Tagging in Distributed Data Lakes

2. Abstract

A system and method for embedding mathematically-derived lineage identifiers within structured and semi-structured data stored in a distributed data lake. The invention enables end-to-end lineage tracking, reproducibility, auditability, and security across different storage and query engines by generating unique lineage keys derived from metadata attributes and transformation signatures.

3. Technical Field

The invention relates to data engineering and governance in distributed systems, specifically to methods for embedding and tracing data lineage through mathematically-generated keys across data lakes and query engines.

4. Background

Traditional data lineage techniques rely on metadata catalogs and manual tracking. These approaches are fragile and often lose fidelity during transformations or cross-system transfers. A deterministic, embedded lineage tagging system solves these gaps, allowing full traceability at the data level itself.

5. Summary of the Invention

The invention proposes a system where each record or batch in a data pipeline is tagged with a unique, mathematically-generated lineage key. This key is derived using a combination of source system identity, ingestion timestamp, transformation identifiers, and data hashes. These tags are embedded directly within Iceberg tables, S3 storage paths, or embedded metadata fields.

Provisional Patent Draft

6. Detailed Description

Lineage Key Generator: Uses deterministic hash functions like SHA256 or UUIDv5 to produce lineage tags.

Integration Points:

- Embedded in data files (e.g., Parquet row field)

- Stored in Iceberg metadata snapshot properties

- Used as part of S3 folder structure

Supports batch- and record-level granularity, is compatible with Starburst, Spark, and Dagster pipelines, and provides cryptographic assurance of data integrity.

7. Claims

1. A method for generating a unique lineage key using metadata attributes and cryptographic hash functions.

2. A system for embedding said key in storage formats (Parquet, JSON, CSV) and catalog layers (Iceberg, Hive).

3. A mechanism to track and verify lineage continuity across transformations using these tags.

4. An architecture that enables lineage-based fault detection and rollback in distributed data pipelines.

8. Use Case Scenarios

- Auditing data sources and transformations in enterprise data lakes.

- Ensuring reproducibility in machine learning pipelines.

- Detecting lineage breaks during data pipeline failures or breaches.

- Simplifying compliance and governance reporting.

Provisional Patent Application

1. Title of Invention

System and Method for Secure, Encrypted Lineage Tagging in Distributed Data Lakes

2. Abstract

A system and method for embedding cryptographically secure, mathematically derived lineage identifiers within distributed data lakes. The approach leverages salted entropy and SHA256 hashing combined with symmetric encryption to ensure traceability, integrity, and tamper resistance of data lineage across various storage layers including Iceberg, object storage, and metadata catalogs.

3. Technical Field

The invention pertains to data engineering and governance, specifically the secure and verifiable tracking of data lineage across distributed processing systems such as Starburst, Apache Iceberg, and S3 object stores.

4. Background of the Invention

Current approaches to data lineage rely heavily on external metadata catalogs and are vulnerable to transformation errors, lack of integration with storage layers, and limited cryptographic assurance. This invention addresses those gaps by embedding lineage directly in the data storage process using deterministic and secure methods.

5. Summary of the Invention

The invention provides a method for deterministic and secure lineage tracking by generating a unique lineage tag for each table or view based on table metadata, timestamps, and transformation stages. The lineage tag is salted and hashed using SHA256, then encrypted using symmetric encryption such as Fernet. This tag is stored either as a column in the table, metadata property in the Iceberg catalog, or S3 object metadata. The

system supports auditability, reproducibility, and cryptographic verification of data history.

6. Detailed Description

1. Lineage Key Generation: Combine schema name, table/view name, ingestion timestamp, and transformation stage.

2. Add Salt (entropy) to the combined string using cryptographic random generator.

3. Compute SHA256 hash of the salted input to generate a unique deterministic key.

4. Encrypt the hash using a symmetric key algorithm (Fernet).

5. Store the encrypted lineage tag in one or more of the following locations:

- Iceberg table snapshot properties
- S3 path/object metadata
- Data table column (`lineage_tag_encrypted`)
- External metadata catalog or audit log

6. Decryption and verification occur during audit or trace operations using the original symmetric key.

7. Claims

1. A method for generating a salted and hashed lineage identifier from table metadata and processing events.

2. A system for encrypting the lineage identifier using a symmetric encryption key.

3. A method of storing the encrypted identifier in distributed data lake layers including Iceberg, S3, and catalogs.

4. A mechanism for verifying data provenance using the decrypted lineage tag.

5. An architecture for integrating secure lineage tagging within ETL pipelines and federated query engines.

8. Use Case Scenarios

- Audit trail tracking across multiple data transformations.

- Compliance-driven environments requiring full data traceability.

- Reproducibility of ML pipelines across environments.

- Tamper-evident logging and rollback support in data governance platforms.

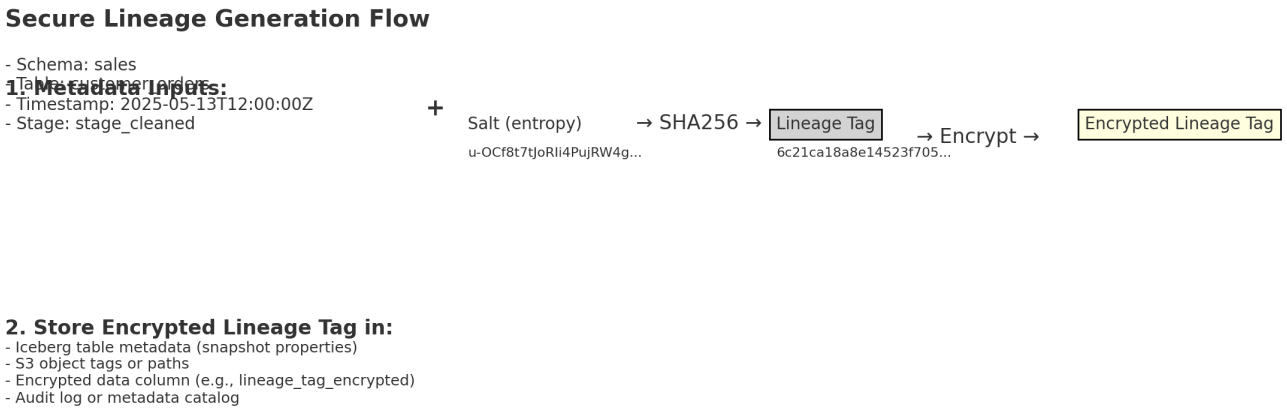
9. System Diagram

Patent Illustration: Secure Lineage Tagging System

Overview

This patent outlines a novel method for creating, encrypting, and embedding data lineage tags within a distributed data lake architecture. The system uses salted SHA256 hashes derived from metadata such as table name, schema, transformation stage, and timestamp to produce a secure lineage tag. This tag is then encrypted using symmetric key cryptography (e.g., Fernet) and stored in metadata layers for traceability.

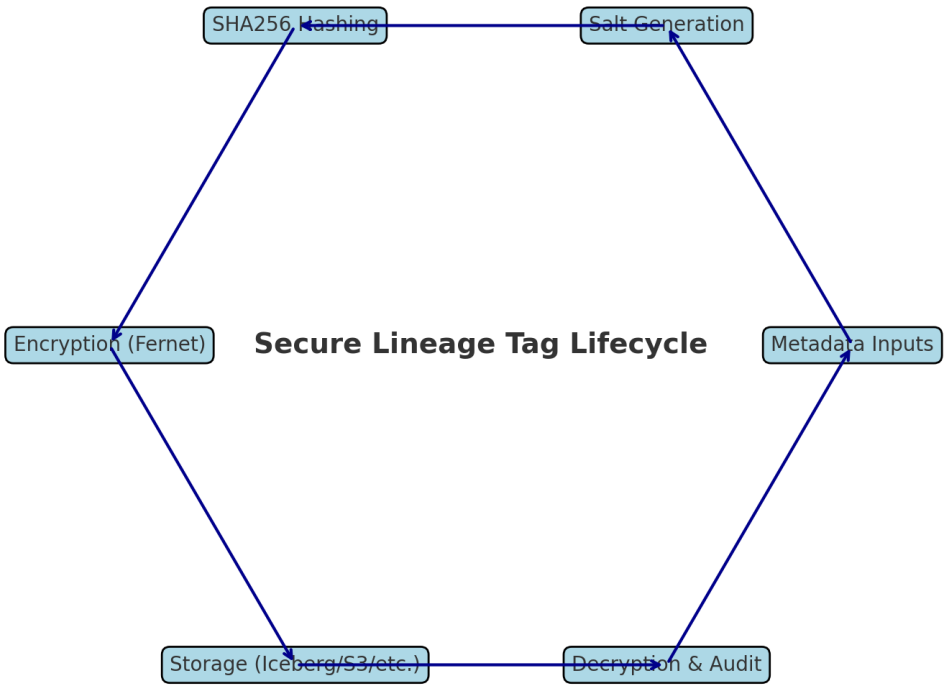
Secure Lineage Flow Diagram



Provisional Patent Application

10. Circular Lifecycle Diagram

The following diagram illustrates the continuous lifecycle of secure lineage tag generation, encryption, storage, and verification. This cyclic approach ensures data integrity, auditability, and governance across all processing stages of the data lake environment.



High-Level Data Lineage Encryption & Decryption Flow

1. Purpose

This document outlines a high-level approach for encrypting and decrypting data lineage tags in distributed data platforms such as Apache Iceberg, Starburst, or object storage like S3.

2. Encryption Flow

Step 1: Generate Lineage Tag

- Use SHA256(schema + table + timestamp + transformation + salt)

Step 2: Generate or Retrieve Encryption Key

- Recommended to use a key from a secure store (AWS Secrets Manager, Vault, etc.)

Step 3: Encrypt the Lineage Tag

- Apply symmetric encryption (e.g., Fernet)

Step 4: Store the Encrypted Tag

- Options:
  - Iceberg table metadata (snapshot properties)
  - A dedicated column in the table (lineage_tag_encrypted)
  - Object metadata in S3
  - External audit logs or metadata catalog

3. Decryption Flow

Step 1: Retrieve Encrypted Tag

- From metadata, table column, or audit store

High-Level Data Lineage Encryption & Decryption Flow

Step 2: Retrieve Encryption Key

- Must match the original encryption key used

Step 3: Decrypt Lineage Tag

- Use symmetric decryption to obtain the original hashed lineage

Step 4: Use for Audit or Traceability

- Reconstruct transformation history
- Validate lineage against expected source

4. Use Case Example

- Table: sales.customer_orders
- Timestamp: 2025-05-13T12:00:00Z
- Transformation: stage_cleaned
- Encrypted Tag: Stored in Iceberg metadata & S3 object tag
- Encryption Key: Managed via AWS Secrets Manager