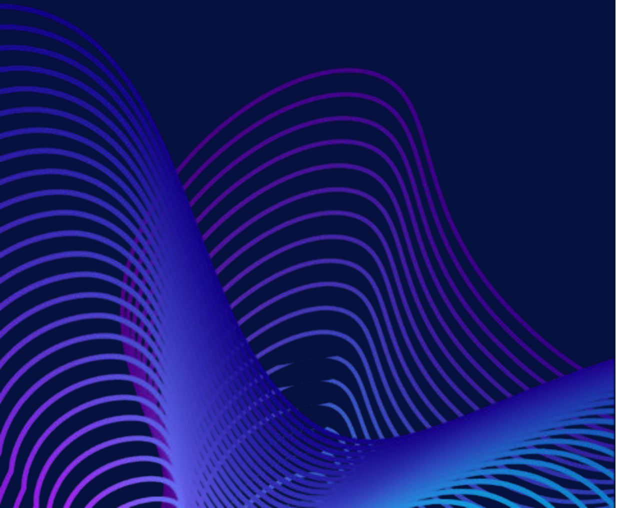
One Stone, Three Birds: Fine-Grained Encryption @ Apache Parquet

Xinli Shang - VP Apache Parquet PMC Chair
Mohammad Islam - Uber Sr. Staff Engineer

Subsurface LIVE

The Cloud Data Lake Conference

Agenda



Encryption, access control and data retention with Apache Parquet fine-grained encryption

1. Context: Why and What
2. Challenges
3. Solutions
4. Take-away

Security & Privacy Building Blocks

Access Control

Retention

Encryption

Fine-Grained Access Control

- Table-level access control is **too broad**.
    - Only handful columns are sensitive.
    - Restricting table level could impact either dev- productivity or wide security access.
- Fine-grained (e.g column-level) access control allows flexibility.

ColName	Col1	Col2	Col3	Col4	Col5
Sensitivity	Low	High	Low	Medium	Low
Access Groups	ALL	G2	ALL	G1, G3	ALL

Data Retention

- Existing deletion removes all columns while only sensitive column need to be deleted.
- Impact: Lose the business value of non-sensitive columns.
- Row-level deletion need to rewrite the data in storage.
- Impact: Expensive and potential error prone.

Encryption

- Encryption of sensitive data is required by some compliance.
- Large scale is challenging and often unnecessary and waste of resources.

Single Solution for All Three Requirements

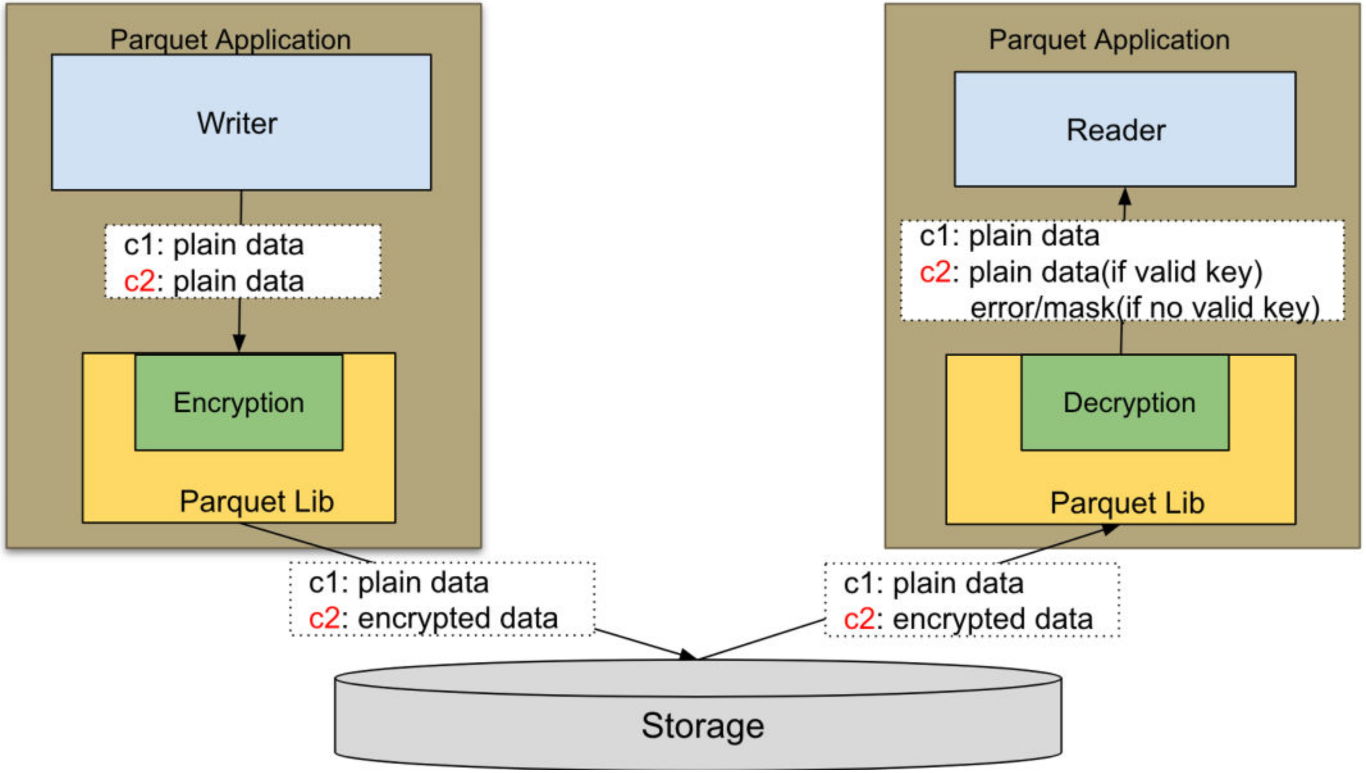
Approach: Utilize encryption to solve all three challenges in Parquet

1. Encrypt only needed columns
2. Control the access at column level with Key Management Service (KMS)
3. Retention on keys instead of data can avoid costly rewriting

Apache Parquet & File Format Encryption

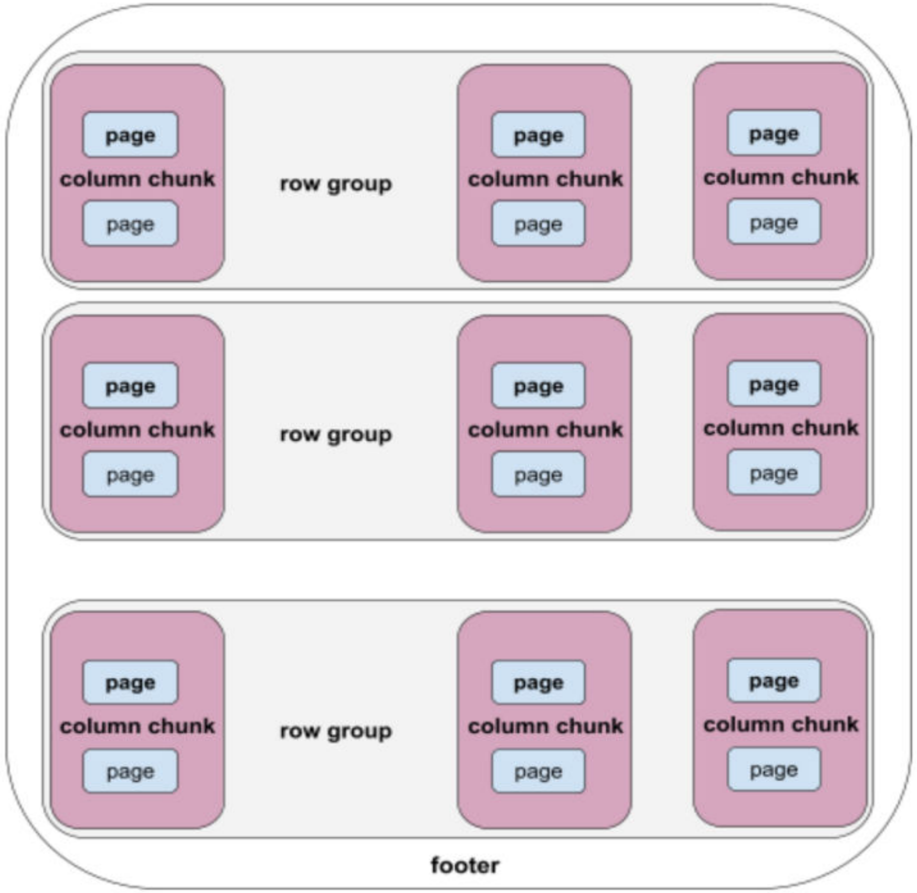
- Choke point to compute engines - Spark, Presto, Hive, Impala etc
- Client side library - protects data at-rest & in-transit
- Widely used columnar file format in big data world

Data Flows of Parquet Encryption



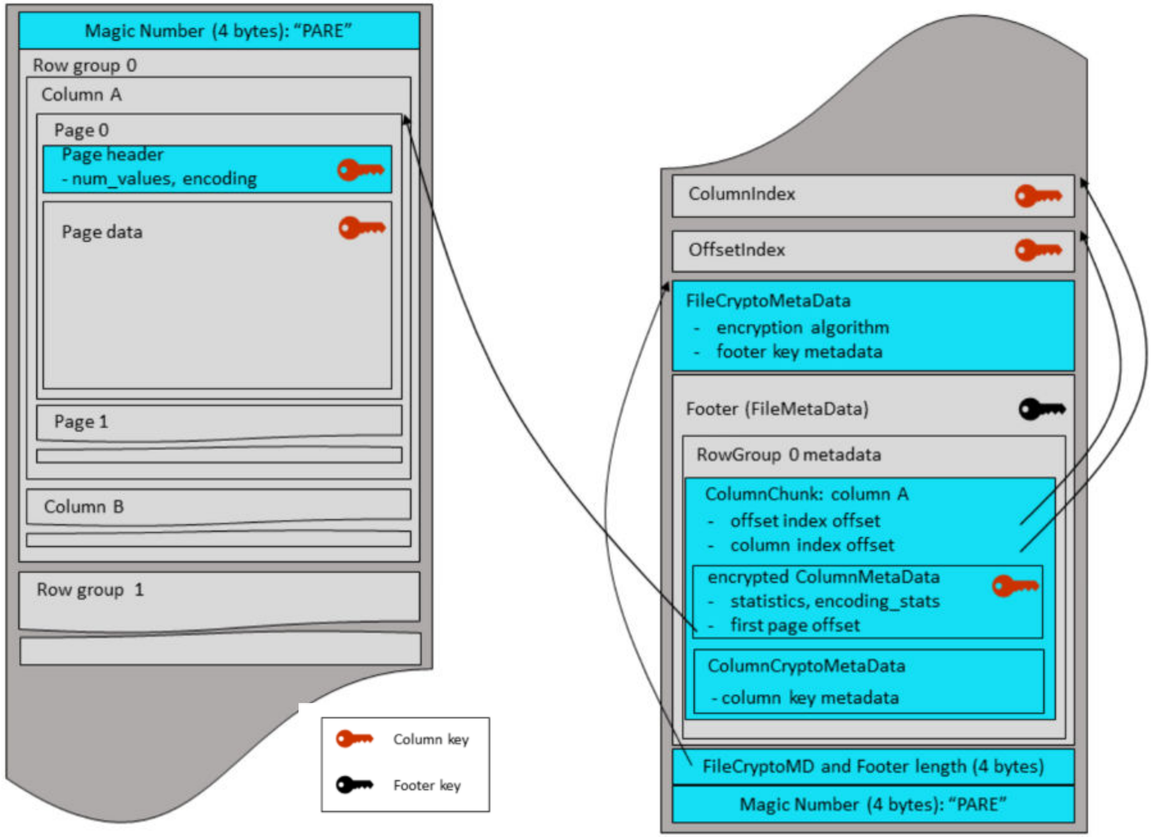
- Encryption details are wrapped in Parquet
- Pluggable encryption/key management
  `EncryptionPropertiesFactory`
- Transparent to data owner and reader

Apache Parquet Format



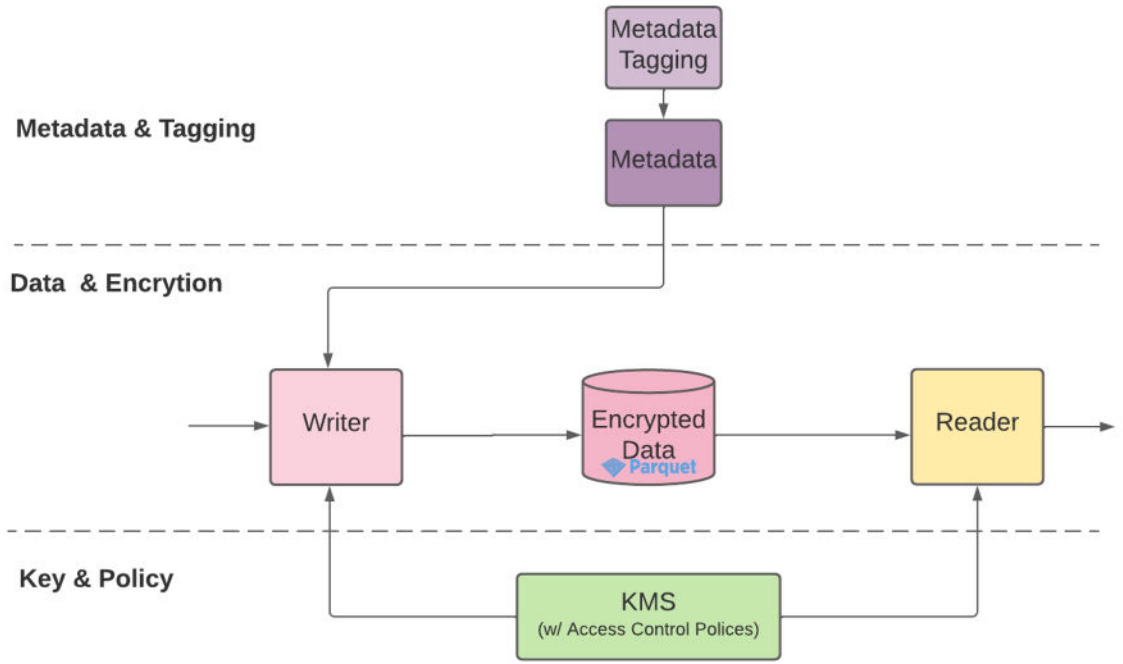
- Modular: footer, row groups, column chunk, page
- Encoding, compression, encryption

Apache Parquet Encryption



- Fine-grained: separate keys for different columns
- Algorithm: AES-CTR and AES-GCM(w/ integration)
- Mode: Plaintext footer, encrypted footer, full encryption

Encryption Automation



- Metadata driven with tagging
- Full automation
- No extra PRCs call introduced in execution
- Policy control on KMS

Performance Overhead

End-2-End Performance Overhead

Write Overhead

$$(T_{encryption\ time} + T_{key\ operation\ time}) / T_{job\ duration}$$

Read Overhead

$$(T_{decryption\ time} + T_{key\ operation\ time}) / T_{job\ duration}$$

Benchmarking: 60% of columns are encrypted, w/ Java 8, AES/CTR

- Write Overhead: **5.7%**
- Read Overhead: **3.7%**

Ground Experiences

Query Engines **Integration**

- Presto
- Spark
- Hive

Plugin as encryption controller

Rewrite **historical** data

- Multiple data format(Parquet/ORC)
- Large scale data
  - 20X encryptor(to be released in 1.13.0)
- Need full automation

User Impact

- Be transparent
  - No interruption
- Reliability of KMS

Key Take-Away

Encryption, access control and retentions are basic security and privacy building blocks

Solve all 3 problems with **single solution** with Parquet encryption

The performance overhead of Parquet encryption is minimal, and Parquet encryption is ready for your production.

Subsurface LIVE

The Cloud Data Lake Conference

Thank You

shangxinli@apache.org

We are hiring!