Encryption at Rest

Rahul Desirazu
August 5, 2022

YFTT
YUGABYTEDB
FRIDAY
TECH TALKS

yugabyteDB

What is Encryption at Rest?

Securing data persisted on each node of the cluster.

Protects against attackers who have read access to the storage on the node.

Does not protect against attackers with write access or access to in-memory state.

High-Level objectives and functional requirements

Enable and disable encryption at a cluster level.

Periodically rotate keys efficiently without a rewrite of the entire data set.

Low performance overhead R/W operations.

Integrate key management with an external Key Management Service (KMS) system.

Never persist encryption keys in plaintext.

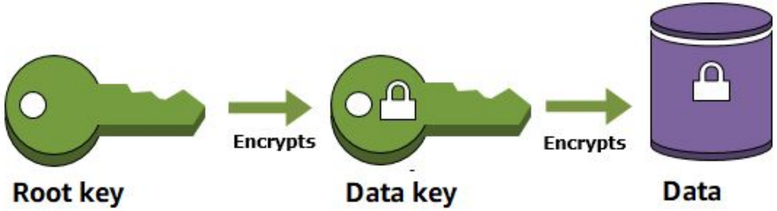
Architecture

Envelope Encryption

Two levels of keys used to implement encryption:

Universe key: Top level symmetric key used to encrypt other keys (see data keys below), one active key per cluster.

Data key: Symmetric key used to encrypt the data. There is one data key generated per flushed data file.



Universe Keys

History of keys stored as an encrypted registry in the database's sys catalog tablet (yb-master).

This registry is encrypted by the latest universe key, which lives plaintext in-memory on the database.

On tablet server startup, yb-master sends it the decrypted registry to seed it with keys it needs for reads and writes.

Universe Key Registry

Typical Universe Key Registry:

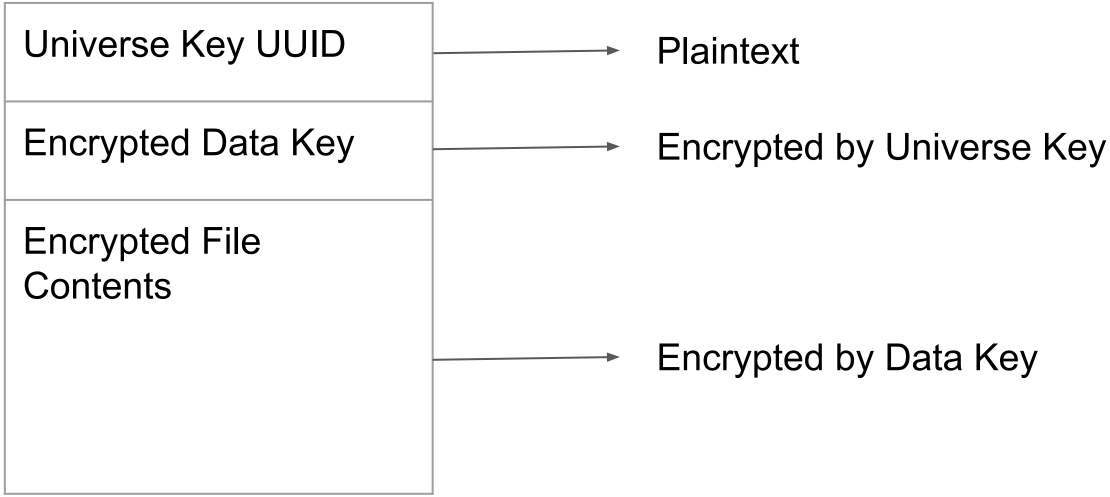
UUID_1	Universe Key 1
UUID_2	Universe Key 2
UUID_3	Universe Key 3
UUID_4	Universe Key 4 (active)

→ Encrypted by Universe Key 4

Data Keys

Generated once per flushed/compacted data (SST) file or Write Ahead Log (WAL) file.

Data File Format:



Universe Key UUID → Plaintext

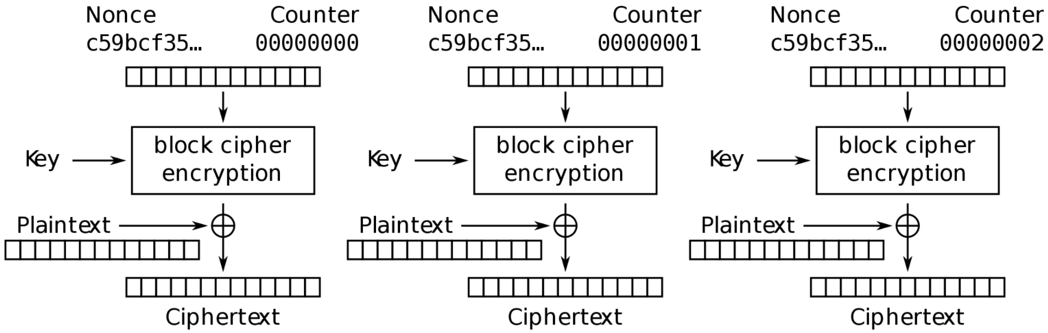
Encrypted Data Key → Encrypted by Universe Key

Encrypted File Contents → Encrypted by Data Key

Encryption Internals

Uses AES CTR-128 mode, which operates on 16 byte blocks at a time using a 16 byte key and 16 byte initialization vector (iv).

Byte *n* in plaintext maps to byte *n* in cipherspace.



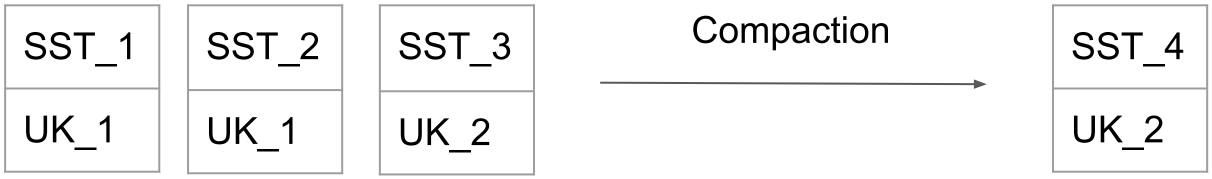
Counter (CTR) mode encryption

Key Rotations

To avoid re-encrypting all data on a key rotation, we do a lazy key rotation:

New data is written using the latest universe key.

Old data is eventually compacted and written with the new key.



Backup Restore

Encrypted backups are stored with with a history of universe key UUIDs.

On restore, platform uses the source universe’s KMS to seed the target with a history of universe keys.

Can restore to a universe with encryption disabled or enabled with a different key.

Demo

Thank You

Join us on Slack: yugabyte.com/slack (#yftt channel)

Star us on Github: github.com/yugabyte/yugabyte-db

YFTT
YUGABYTEDB
FRIDAY
TECH TALKS

yugabyteDB