# **Hadoop HDFS Encryption Components – Detailed Overview**

HDFS **Transparent Data Encryption (TDE)** secures **data at rest** by encrypting files within **HDFS Encryption Zones**. It ensures that **only authorized users and applications can access encrypted data** while maintaining **seamless integration with Hadoop services**.

## **🔹 Key Components of HDFS Encryption**

| **Component** | **Description** |
| --- | --- |
| **Encryption Zones (EZs)** | Special directories where all files are automatically encrypted. |
| **Key Management Server (KMS)** | Manages encryption keys and access control. |
| **Encryption Keys (EZK, DEK, EDEK)** | Different keys used for securing data at multiple levels. |
| **HDFS Client Encryption Handling** | Encrypts and decrypts data at the **client-side**. |
| **Access Control & Policies** | Enforced through **Ranger KMS** or **ACLs** to restrict unauthorized access. |

## **🔹 1. Encryption Zones (EZs)**

HDFS **Encryption Zones** are directories where **all data is automatically encrypted** before being stored.

✔ **All files written to an Encryption Zone are encrypted**.  
✔ **Prevents unauthorized access, even if HDFS DataNodes are compromised**.  
✔ **Transparent to applications – no code changes required**.

**🔹 Example: Creating an Encryption Zone**

bash

CopyEdit

hdfs dfs -mkdir /secure\_zone

hdfs crypto -createZone -keyName my-secure-key -path /secure\_zone

**🔹 Verify Encryption Zones**

bash

CopyEdit

hdfs crypto -listZones

✅ Output:

bash

CopyEdit

/secure\_zone my-secure-key

🔹 **Confirms /secure\_zone is encrypted with my-secure-key**.

## **🔹 2. Key Management Server (KMS)**

**KMS is responsible for managing encryption keys securely.**It ensures **only authorized users** can encrypt and decrypt files.

✔ Stores **Encryption Zone Keys (EZKs)** securely.  
✔ Generates **Data Encryption Keys (DEKs)** dynamically.  
✔ Enforces **Access Control Lists (ACLs)** via **Ranger KMS**.

**🔹 Example: Creating an Encryption Key in KMS**

bash

CopyEdit

hadoop key create my-secure-key -provider kms://http@localhost:16000/kms

✅ **Registers the encryption key in KMS**.

## **🔹 3. Encryption Keys Used in HDFS TDE**

| **Key Type** | **Function** | **Storage Location** |
| --- | --- | --- |
| **Encryption Zone Key (EZK)** | Master key for encrypting DEKs | **Stored in KMS** |
| **Data Encryption Key (DEK)** | Encrypts file content | **Generated dynamically** |
| **Encrypted Data Encryption Key (EDEK)** | DEK encrypted using EZK | **Stored in HDFS metadata** |

✔ **HDFS only stores encrypted versions of keys (EDEKs), never raw DEKs**.

**🔹 Example: Retrieving Encryption Key Metadata**

bash

CopyEdit

hadoop key metadata my-secure-key -provider kms://http@localhost:16000/kms

✅ Displays **key versions, creation time, and ACLs**.

## **🔹 4. HDFS Client Encryption Handling**

HDFS clients **handle encryption and decryption** on the **client-side**.

**🔹 How it Works:** 1️⃣ **Client requests a file from an Encryption Zone**2️⃣ **HDFS retrieves the EDEK from metadata**3️⃣ **Client sends EDEK to KMS for decryption**4️⃣ **KMS returns DEK to the client (if authorized)**5️⃣ **Client decrypts the file and reads plaintext data**

🔹 **HDFS DataNodes never see unencrypted data**, ensuring security.

## **🔹 5. Access Control & Security Policies**

HDFS encryption relies on **Ranger KMS or Hadoop ACLs** for access control.

✔ **Restrict who can read encrypted files**.  
✔ **Control which users can decrypt EDEKs**.  
✔ **Prevent unauthorized key access in KMS**.

### **🔹 Example: Granting User Access to Encryption Keys in Ranger KMS**

Modify **kms-acls.xml**:

xml

CopyEdit

<property>

<name>hadoop.kms.acl.my-secure-key.DECRYPT\_EEK</name>

<value>finance\_group</value>

</property>

✅ **Only finance\_group users can decrypt files in /secure\_zone**.

**🔹 Test Unauthorized Access**

bash

CopyEdit

sudo -u unauthorized\_user hdfs dfs -cat /secure\_zone/confidential.txt

❌ **Expected Output:**

pgsql

CopyEdit

Permission denied: User unauthorized\_user does not have access to encryption key.

✔ **Ensures that unauthorized users cannot read encrypted data**.

## **🔹 6. Audit Logging & Monitoring**

✔ **Tracks who accessed encryption keys**.  
✔ **Logs encryption and decryption requests**.  
✔ **Helps meet compliance requirements (e.g., GDPR, PCI DSS)**.

**🔹 Enable HDFS Audit Logging in Ranger KMS** Modify **ranger-kms-site.xml**:

xml

CopyEdit

<property>

<name>xasecure.audit.destination.hdfs</name>

<value>true</value>

</property>

Restart Ranger KMS:

bash

CopyEdit

sudo systemctl restart ranger-kms

✅ **All key access events are now logged**.

## **🔹 Summary**

| **Component** | **Purpose** |
| --- | --- |
| **Encryption Zones (EZs)** | Encrypts all data stored inside the directory. |
| **Key Management Server (KMS)** | Manages encryption keys securely. |
| **Encryption Keys (EZK, DEK, EDEK)** | Ensures secure encryption and decryption. |
| **HDFS Client Encryption Handling** | Encrypts and decrypts data at the **client-side**. |
| **Access Control & Policies** | Restricts key access and file decryption using **Ranger KMS**. |
| **Audit Logging & Monitoring** | Tracks key access and decryption events. |

✔ **HDFS Transparent Data Encryption (TDE) protects sensitive data at rest**.  
✔ **Key security is enforced through KMS and Ranger policies**.  
✔ **Only authorized users can decrypt encrypted files**.