# Hadoop Encrypted Shuffle

## Table of contents

## 1. Introduction

The Encrypted Shuffle capability allows encryption of the MapReduce shuffle using HTTPS and with optional client authentication (also known as bi-directional HTTPS, or HTTPS with client certificates). It comprises:

- A Hadoop configuration setting for toggling the shuffle between HTTP and HTTPS.
- A Hadoop configuration settings for specifying the keystore and truststore properties (location, type, passwords) used by the shuffle service and the reducers tasks fetching shuffle data.
- A way to re-load truststores across the cluster (when a node is added or removed).

## 2. Configuration

**core-site.xml** Properties

To enable encrypted shuffle, set the following properties in **core-site.xml** of all nodes in the cluster:

- **hadoop.ssl.require.client.cert**. Default:**false**. Whether client certificates are required
- **hadoop.ssl.hostname.verifier**. Default:**DEFAULT**. The hostname verifier to provide for HttpsURLConnections. Valid values are:**DEFAULT**,**STRICT**,**STRICT_I6**, **DEFAULT_AND_LOCALHOST** and **ALLOW_ALL**
- **hadoop.ssl.keystores.factory.class**. Default: **org.apache.hadoop.security.ssl.FileBasedKeyStoresFactory** . The KeyStoresFactory implementation to use
- **hadoop.ssl.server.conf**. Default:**ss-server.xml**. Resource file from which ssl server keystore information will be extracted. This file is looked up in the classpath, typically it should be in Hadoop conf/ directory
- **hadoop.ssl.client.conf**. Default:**ss-client.xml**. Resource file from which ssl server keystore information will be extracted. This file is looked up in the classpath, typically it should be in Hadoop conf/ directory

**IMPORTANT:** Currently requiring client certificates should be set to false. Refer the Client Certificates section for details.

**IMPORTANT:** All these properties should be marked as final in the cluster configuration files.

**Example:**

```
        ...
        <property>
```

```
            <name>hadoop.ssl.require.client.cert</name>
            <value>false</value>
            <final>true</final>
          </property>

          <property>
            <name>hadoop.ssl.hostname.verifier</name>
            <value>DEFAULT</value>
            <final>true</final>
          </property>

          <property>
            <name>hadoop.ssl.keystores.factory.class</name>
            <value>org.apache.hadoop.security.ssl.FileBasedKeyStoresFactory
            </value>
            <final>true</final>
          </property>

          <property>
            <name>hadoop.ssl.server.conf</name>
            <value>ssl-server.xml</value>
            <final>true</final>
          </property>

          <property>
            <name>hadoop.ssl.client.conf</name>
            <value>ssl-client.xml</value>
            <final>true</final>
          </property>
          ...
```

**mapred-site.xml** Properties

To enable encrypted shuffle, set the following property in **mapred-site.xml** of all nodes in the cluster:

- **mapreduce.shuffle.ssl.enabled**. Default:**false**. Whether encrypted shuffle is enabled

**IMPORTANT:** All these properties should be marked as final in the cluster configuration files.

**Example:**

```
          ...
          <property>
            <name>mapreduce.shuffle.ssl.enabled</name>
            <value>true</value>
          </property>
          ...
```

The Linux container executor should be set to prevent job tasks from reading the server

keystore information and gaining access to the shuffle server certificates.

Refer to Hadoop Kerberos configuration for details on how to do this.

## 3. Keystore and Truststore Settings

Currently **FileBasedKeyStoresFactory** is the only **KeyStoresFactory** implementation. The **FileBasedKeyStoresFactory** implementation uses the following properties, in the **ssl-server.xml** and **ssl-client.xml** files, to configure the keystores and truststores.

**ssl-server.xml** (Shuffle server) Configuration:

The mapred user should own the **ssl-server.xml** file and have exclusive read access to it.

- **ssl.server.keystore.type**. Default **jks**. Keystore file type
- **ssl.server.keystore.location**. Default **NONE**. Keystore file location. The mapred user should own this file and have exclusive read access to it
- **ssl.server.keystore.password**. Default **NONE**. Keystore file password
- **ssl.server.truststore.type**. Default **jks**. Truststore file type
- **ssl.server.truststore.location**. Default **NONE**. Truststore file location. The mapred user should own this file and have exclusive read access to it
- **ssl.server.truststore.password**. Default **NONE**. Truststore file password
- **ssl.server.truststore.reload.interval**. Default **10**. Truststore reload interval, in milliseconds

**Example:**

```
<configuration>

  <!-- Server Certificate Store -->
  <property>
    <name>ssl.server.keystore.type</name>
    <value>jks</value>
  </property>
  <property>
    <name>ssl.server.keystore.location</name>
    <value>${user.home}/keystores/server-keystore.jks</value>
  </property>
  <property>
    <name>ssl.server.keystore.password</name>
    <value>serverfoo</value>
  </property>

  <!-- Server Trust Store -->
  <property>
    <name>ssl.server.truststore.type</name>
    <value>jks</value>
```

```
  </property>
  <property>
    <name>ssl.server.truststore.location</name>
    <value>${user.home}/keystores/truststore.jks</value>
  </property>
  <property>
    <name>ssl.server.truststore.password</name>
    <value>clientserverbar</value>
  </property>
  <property>
    <name>ssl.server.truststore.reload.interval</name>
    <value>10000</value>
  </property>
</configuration>
```

**ssl-client.xml** (Reducer/Fetcher) Configuration:

The mapred user should own the **ssl-server.xml** file and it should have default permissions.

- **ssl.client.keystore.type**. Default **jks**. Keystore file type
- **ssl.client.keystore.location**. Default **NONE**. Keystore file location. The mapred user should own this file and have exclusive read access to it
- **ssl.client.keystore.password**. Default **NONE**. Keystore file password
- **ssl.client.truststore.type**. Default **jks**. Truststore file type
- **ssl.client.truststore.location**. Default **NONE**. Truststore file location. The mapred user should own this file and have exclusive read access to it
- **ssl.client.truststore.password**. Default **NONE**. Truststore file password
- **ssl.client.truststore.reload.interval**. Default **10**. Truststore reload interval, in milliseconds

**Example:**

```
<configuration>

  <!-- Clietn Certificate Store -->
  <property>
    <name>ssl.client.keystore.type</name>
    <value>jks</value>
  </property>
  <property>
    <name>ssl.client.keystore.location</name>
    <value>${user.home}/keystores/client-keystore.jks</value>
  </property>
  <property>
    <name>ssl.client.keystore.password</name>
    <value>clientfoo</value>
  </property>

  <!-- Client Trust Store -->
  <property>
```

```
    <name>ssl.client.truststore.type</name>
    <value>jks</value>
  </property>
  <property>
    <name>ssl.client.truststore.location</name>
    <value>${user.home}/keystores/truststore.jks</value>
  </property>
  <property>
    <name>ssl.client.truststore.password</name>
    <value>clientbar</value>
  </property>
  <property>
    <name>ssl.client.truststore.reload.interval</name>
    <value>10000</value>
  </property>
</configuration>
```

## 4. Activating Encrypted Shuffle

When you have made the above configuration changes, activate Encrypted Shuffle by re-starting all TaskTrackers.

**IMPORTANT:** Using encrypted shuffle will incur in a significant performance impact. Users should profile this and potentially reserve 1 or more cores for encrypted shuffle.

## 5. Client Certificates

Using Client Certificates does not fully ensure that the client is a reducer task for the job. Currently, Client Certificates (their private key) keystore files must be readable by all users submitting jobs to the cluster. This means that a rogue job could read such those keystore files and use the client certificates in them to establish a secure connection with a Shuffle server. However, unless the rogue job has a proper JobToken, it won't be able to retrieve shuffle data from the Shuffle server. A job, using its own JobToken, can only retrieve shuffle data that belongs to itself.

## 6. Reloading Truststores

By default the truststores will reload their configuration every 10 seconds. If a new truststore file is copied over the old one, it will be re-read, and its certificates will replace the old ones. This mechanism is useful for adding or removing nodes from the cluster, or for adding or removing trusted clients. In these cases, the client or TaskTracker certificate is added to (or removed from) all the truststore files in the system, and the new configuration will be picked up without you having to restart the TaskTracker

## 7. Debugging

**NOTE:** Enable debugging only for troubleshooting, and then only for jobs running on small amounts of data. It is very verbose and slows down jobs by several orders of magnitude. (You might need to increase mapred.task.timeout to prevent jobs from failing because tasks run so slowly.)

To enable SSL debugging in the reducers, set **-Djavax.net.debug=all** in the **mapreduce.reduce.child.java.opts** property; for example:

```
<property>
  <name>mapred.reduce.child.java.opts</name>
  <value>-Xmx-200m -Djavax.net.debug=all</value>
</property>
```

You can do this on a per-job basis, or by means of a cluster-wide setting in the **mapred-site.xml** file.

To set this property in TaskTracker, set it in the **hadoop-env.sh** file:

```
HADOOP_OPTS="-Djavax.net.debug=all $HADOOP_OPTS"
```