# Optimize Hbase RPC encryption performance

## Introduction

Hbase RPC encryption is enabled by setting “hbase.rpc.protection” to "privacy". With the token authentication, it utilized DIGEST-MD5 mechanisms for secure authentication and data protection. For DIGEST-MD5, it uses DES, 3DES or RC4 to do encryption and it is very slow, especially for Scan. This will become the bottleneck of the RPC throughput.

Apache Commons Crypto is a cryptographic library optimized with AES-NI. It provides Java API for both cipher level and Java stream level. Developers can use it to implement high performance AES encryption/decryption with the minimum code and effort. Compare with the current implementation of org.apache.hadoop.hbase.io.crypto.aes.AES, Crypto supports both JCE Cipher and OpenSSL Cipher which is better performance than JCE Cipher. User can configure the cipher type and the default is JCE Cipher.

In this design, we will implement RPC encryption with Crypto and it won’t be the bottleneck any more.

## Goals

1. Support using Crypto to do encryption for Hbase RPC.

2. Compatibility: the feature can be configured and won’t block other function.

3. Performance: with the token authentication and data protection, the RPC throughput should be improved more than 2.5X based on the result of YCSB with Scan test.

## Design

Create SaslAES class to wrap the Crypto cipher and it’s responsible for encryption/decryption with Crypto cipher (Supports JCE Cipher and OpenSSL Cipher).

To initialize cipher, Hbase client and server will negotiate cipher option after SASL handshake. To ensure the compatibility, client and server will negotiate the Crypto cipher if “hbase.crypto.sasl.encryption.aes.enabled” is true, and SaslAES instance will be created for both server and client. During the negotiation:

1. Client will send the transformation to server for the Crypto cipher
2. Server will generate the key and IV, then create the instance of SaslAES with them and transformation from client. At last, server return the transformation, key, IV to client. The negotiation is finished in server side.
3. Client will create the instance of SaslAES with the transformation, key, IV from server and the negotiation is finished in client side.

The following pictures show the steps of Crypto cipher negotiation and encryption/decryption with Crypto cipher:

