The Performance Testing of Diceros/SunJCE

[Abstract 2](#_Toc390771260)

[Introduction 2](#_Toc390771261)

[Test Platform 2](#_Toc390771262)

[Test Methodology 2](#_Toc390771263)

[Measurement Results–Standard AES Algorithm 3](#_Toc390771264)

# Abstract

Advanced Encryption Standard (AES) is the ﬁrst cryptographic standard aroused as a result of public competition that was established by U.S. National Institute of Standards and Technology. This paper represents a study which compares performance of well known cryptographic provider - SunJCE in relation to our own specialized implementations of AES algorithm – Diceros. The paper aims to determine advantages between the two implementations. Having compared the these implementations, our evaluation results show that Diceros gave pretty better performance results than SunJCE in encryption and decryption. It should be noted that the results presented in this study will show some advantages of Diceros related not only to algorithm speed, but also to possibilities for further analysis of the algorithm.

# Introduction

Diceros is a sub-project of Rhino project (https://github.com/intel-hadoop/project-rhino/) which focuses on providing a hardware accelerated JCE provider. Initial effort includes:

* AES-NI enabled AES/CTR/NOPADDING
* AES/CBC/PKCS5PADDING and AES/CBC/NOPADDING
* Hardware based true random generator (DRNG)

This paper represents an empirical study which compares performance of massive and well known cryptographic provider in relation to Diceros implementations of AES algorithm in Java programming language. In the paper, we will further compare these results with speed measurements of an experiment with AES algorithm extensions between the key size of 128 bits and 256 bits. As a reference for measuring, we will use well-known AES implementations, which are parts of the large cryptographic provider – SunJCE. Comprised in our evaluation, we had the cryptographic providers and the length of the keys used in the experiments are:

* SunJCE use 128 and 256 bits key with AES/CTR/NOPADDING,AES/CBC/NOPADDING and AES/CBC/PKCS5PADDING modes in encryption/decryption
* Diceros use 128 and 256 bits key with AES/CTR/NOPADDING,AES/CBC/NOPADDING and AES/CBC/PKCS5PADDING modes in encryption/decryption

# Test Platform

As a test platform was used a computer with Intel(R) Xeon(R) CPU E5-2680 processor at 2.70 GHz (new AES set of instructions - AES-NI enable), with 128G RAM and with the CentOS 6.4 operating system. As a development environment we used Java SE Development Kit 7u45 and Java Cryptography Extension for Linux

# Test Methodology

To achieve the highest test results precision, this paper uses the Java Microbenchmark Harness (JMH) framework to implement the performance testing. Besides using this testing framework, each individual implementation testing was given the same conditions in regard to processor and memory. Each particular implementation was evaluated using the same test platform as described above. All tests were conducted by consecutive repetition of measurements on dataSize in 16B-128KB size.

JMH is a Java harness for building, running, and analyzing nano/micro/milli/macro benchmarks written in Java and other languages targeting the JVM. JMH is Maven-driven, hence having Maven installed will bring the best experience.JMH allowed us to deeply understand implementation details, which were impacting execution and concurrency of our new implementation. JMH has only 2 requirements:

* You need jmh-core maven dependency
* You need to annotate test methods with @GenerateMicroBenchmark annotation

Annotate the required methods with @GenerateMicroBenchmark. Performance Testing JMH code with annotation "@GenerateMicroBenchmark" will looks like below:

@GenerateMicroBenchmark

public void encryptPerfTest(){

...

}

@GenerateMicroBenchmark

public void decryptPerfTest(){

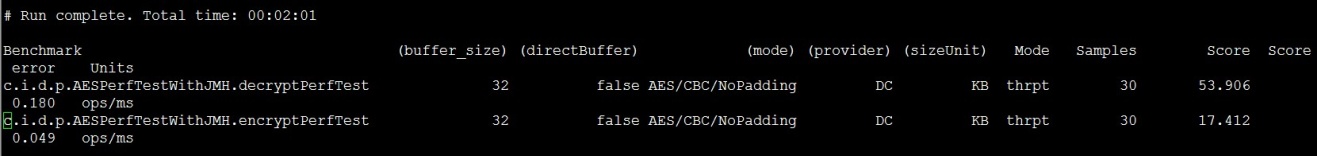
...

}

When finish testing code and install with maven, performance testing as following command:

Java -jar microbenchmarks.jar (.\*encryptPerfTest|.\*encryptPerfTest) -wi 20 -i 30 -f 1 -p provider=(SunJCE|Diceros) -p mode=(AES/CTR/NoPadding|AES/CBC/PKCS5Padding|AES/CBC/NoPadding) -p directBuffer=(true|false) -p buffer\_size=(18B-128KB) -p sizeUnit=(B|KB)

Here are the benchmark output details with mode of operations Throughput are shown below:



# Measurement Results–Standard AES Algorithm

Hereby we set out the measurement results, with the aim to rank our implementations – Diceros in comparison to large cryptographic provider – SunJCE. We implement performance testing with ByteArray and ByteBuffer as input between Diceros and SunJCE.

Figure1: 128BIT encryption results SunJCE VS Diceros

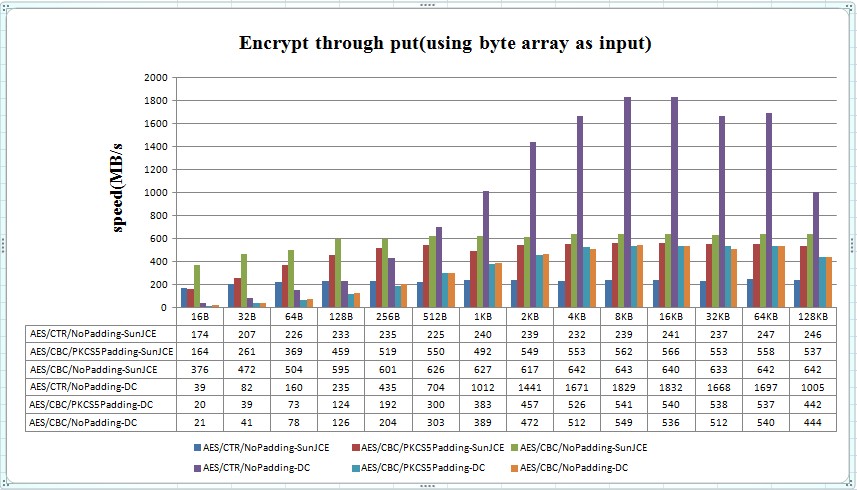


Figure2: 128BIT decryption results SunJCE VS Diceros

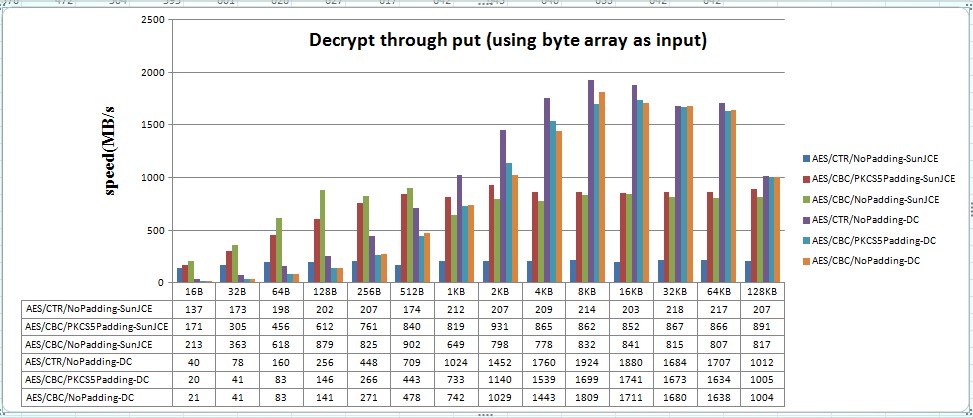


Figure3: 128BIT encryption results SunJCE VS Diceros

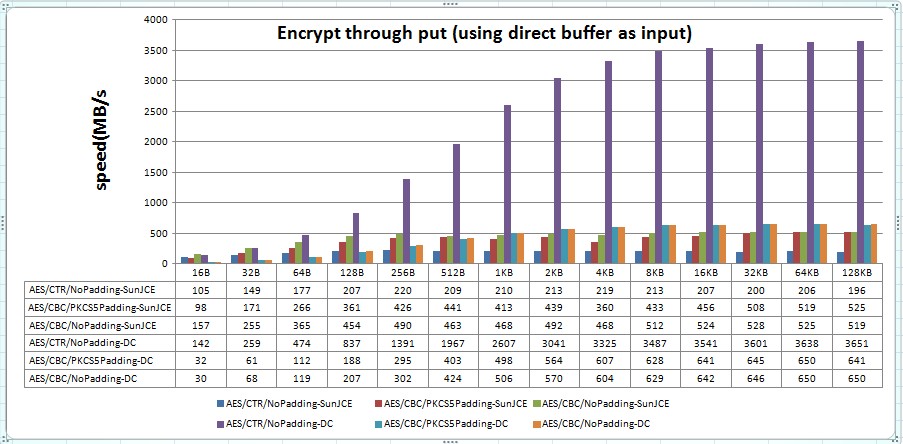
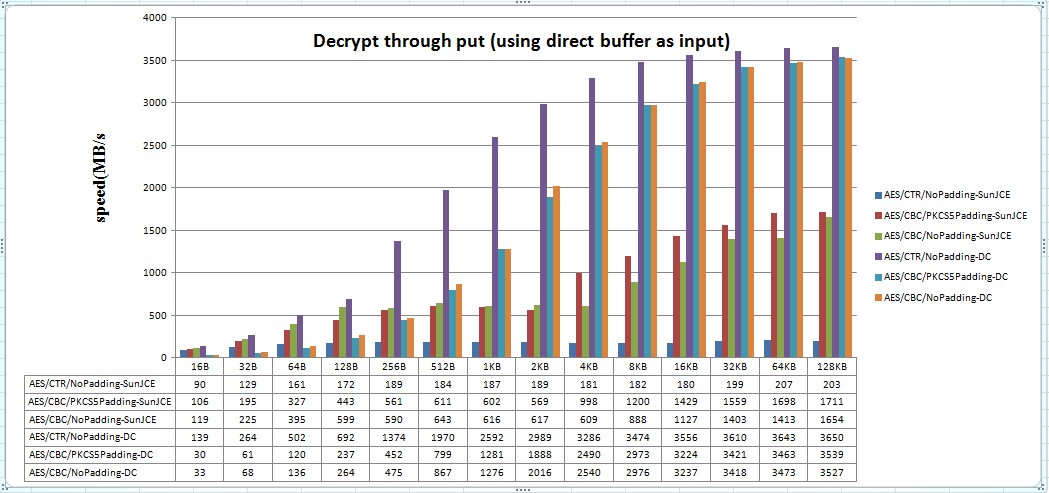


Figure4: 128BIT decryption results SunJCE VS Diceros



The above diagrams show the results of measurement encrypt/decrypt speed, Diceros encrypts data slightly slower than SunJCE with aes/cbc/pkcs5padding and aes/cbc/nopadding mode, but greatly faster than SunJCE with aes/ctr/nopadding mode (because AES new instructions enable). Diceros decrypts date greatly faster than SunJCE with any decrypt cipher mode.