An Industry Standard Benchmark Consortium

DENBench™ Version 1.0 — Benchmark Name: AES

Highlights

- **Benchmarks the Rijndael Algorithm of the Advanced Encryption Standard (AES) algorithm, named for creators Joan Daemen and Vincent Rijmen**
- **Roundtrip implementation plus integration of the FIPS test assures accuracy**
- **A component of the EEMBC Cryptography sub-suite**
- **AES tends to replace DES and Triple-DES as the preferred encryption algorithm for maximum-security applications in governmental, HDTV, satellite, and vital data-security applications**

Application and Restrictions

The Advanced Encryption Stand (AES) benchmark provides an indication of the potential performance of a microprocessor or Digital Signal Processor (DSP) subsystem doing AES cryptographic encryptions and decryptions. The AES cipher is used in numerous cryptographic protocols, including Transport Layer Security (TLS), Secure Socket Layer, (SSL), Secure Shell, (SSH), and Internet Protocol Security (IPSEC). AES is a royalty-free Federal Information Processing Standards (FIPS) approved standard intended to ultimately replace Digital Encryption Standard (DES). The EEMBC benchmark and its source code are subject to the following restrictions:

> *Joan Daemen and Vincent Rijmen, the developers of the Rijndael algorithm, submitted this software package to the National Institute of Standards and Technology (NIST) during the Advanced Encryption Standard (AES) development effort. This software is distributed in compliance with export regulations (see below), and it is intended for non-commercial use only. NIST does not support this software and does not provide any guarantees or warranties as to its performance, fitness for any particular application, or validation under the Cryptographic Module Validation Program (CMVP) http://csrc.nist.gov/cryptval. NIST does not accept any liability associated with its use or misuse. This software is provided as-is. By accepting this software the user agrees to the terms stated herein.*

The EEMBC AES Benchmark Software is subject to the following Export Restrictions (exportation from the United States of America to non-USA countries): *Implementations of cryptography are subject to United States Federal Government export controls. Export controls on commercial encryption products are administered by the Bureau of Export Administration (BXA) http://www.bxa.doc.gov/Encryption/ in the U.S. Department of Commerce. Regulations governing exports of encryption are found in the Export Administration Regulations (EAR), 15 C.F.R. Parts 730-774. Compliance with export restrictions is the responsibility of each individual EEMBC member, not EEMBC, Inc. itself.*

Benchmark Description

Rijndael is an iterated block cipher with a variable block length and a variable key length. The block length and the key length can be independently specified to 128, 192, or 256 bits. The EEMBC AES benchmark implements all three of these key lengths for each iteration. Like most cryptographic functions, there is an array (or "block") that is subjected to multiple transformations. This benchmark performs 1000 Monte Carlo Tests (MCT) for 16 passes, and does a complete round-trip encryption, followed by decryption, to verify correctness. Implementing the FIPS tests inside the code itself further enhances correctness. The code faithfully implements the Wide Trail Strategy to deflect against layer attacks, and implements all three layers: linear-mixing, non-linear layer, and key addition layer.

Unlike the other EEMBC cryptography benchmarks (which were created based on the SSLEAY 0.9 project), the EEMBC AES benchmark was created from the baseline source code and specification by Vincent Rijmen and Joan Daemon found at http://csrc.nist.gov/CryptoToolkit/aes/. It supports the FIPS test found at http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

The data that is supplied is proprietary to EEMBC. In addition, EEMBC has its own, secret data that is used to double-check accuracy of implementation.

Analysis of Computing Resources

The benchmark is computationally challenging: addition, multiplication, extensive use of division, bit shifting, matrix math, bitwise operators such as XOR, and other operators are used. It is implemented in integer math. This benchmark is almost exclusively CPU bound, and the quality of the math library as well as memory library has an effect on performance. Memory moves are performed repeatedly, so optimized C library `mem*` functions would improve performance, but without overwhelming the basic math computations. Sophisticated superscalar architectures scheduled by sophisticated compilers (or assembly language implementations) can take advantage of some parallelism.