Analysis of the Intel AES-NI Special Instruction Set

Shawn Wilkinson

ECE646 Fall 2015

Overview

- Background
- AES-NI Basics
- Key and Block Sizes
- New Instructions
- Microprocessor Operations
- Other Benefits
- Advertised Performance Increases
- Tested Performance Increases

Background

- Purpose:
  - The AES-NI instruction set extensions are used to optimize encryption and decryption algorithms on select Intel and AMD processors.
- History:
  - NIST selected Rijndael to become AES in Oct, 2000.
  - Intel announced AES-NI in 2008 and released supported CPUs late 2010.
- Applications:
  - Bulk encryption
    - ECB, CBC, CTR
  - Data Authentication
    - CBC-MACs, CMAC
  - Random Number Generation
    - CTR-DRBG

AES-NI Basics

- Typical software implementations of AES require multiple steps for each round of encryption.
  - Calculate every step of the AES key schedule
  - Substitute S-boxes
  - Shift rows
  - Mix the columns
  - XOR the round key
- Using AES-NI, only one instruction is required to perform one round of AES encryption or decryption.
- Pipelining can be utilized for modes of operation that allow it, allowing very large performance increases.
- For modes of operation that must be done sequentially, AES-NI can only help with one encryption or decryption step at a time, but it still greatly increases the speed because this encryption or decryption operation is the slowest part of CBC mode.

Key and Block sizes

- Standard key lengths: 128, 192, 256
- Standard block size of 128 bits
  - Not limited to the above. Any block size that is a multiple of 32 bits allowed based on original Rijndael specifications.

- AES-128: 10 rounds (40 steps)
- AES-192: 12 rounds (48 steps)
- AES-256: 14 rounds (56 steps)

New Instructions

Encryption/Decryption:

AESENC	Perform one round of an AES encryption flow
AESENCLAST	Perform the last round of an AES encryption flow
AESDEC	Perform one round of an AES decryption flow
AESDECLAST	Perform the last round of an AES decryption flow

Key Expansion:

AESKEYGENASSIST	Assist in AES round key generation
AESIMC	Assist in AES Inverse Mix Columns

Miscellaneous:

PCLMULQDQ	Carryless multiply (CLMUL)

Microprocessor Operations: Encryption using AES-NI

- AESENC xmm1, xmm2/m128
  - Tmp := xmm1
  - RoundKey :=xmm2/m128
  - Tmp := ShiftRows (Tmp)
  - Tmp := SubBytes (Tmp)
  - Tmp := MixColumns (Tmp)
  - xmm1:= Tmp xor RoundKey
- AESENCLAST xmm1, xmm2/m128
  - Tmp := xmm1
  - RoundKey :=xmm2/m128
  - Tmp := ShiftRows (Tmp)
  - Tmp := SubBytes (Tmp)
  - xmm1:= Tmp xor RoundKey

Microprocessor Operations: Decryption using AES-NI

- AESDEC xmm1, xmm2/m128
  - Tmp := xmm1
  - RoundKey :=xmm2/m128
  - Tmp := InvShiftRows (Tmp)
  - Tmp := InvSubBytes (Tmp)
  - Tmp := InvMixColumns (Tmp)
  - xmm1:= Tmp xor RoundKey
- AESDECLAST xmm1, xmm2/m128
  - Tmp := xmm1
  - RoundKey :=xmm2/m128
  - Tmp := InvShiftRows (Tmp)
  - Tmp := InvSubBytes (Tmp)
  - xmm1:= Tmp xor RoundKey

Key Expansion

- AESKEYGENASSIST xmm1, xmm2/m128, imm8
  - Tmp := xmm2/m128
  - RCON[31-8] := 0; RCON[7-0] := imm8;
  - X3[31-0] := Tmp[127-96]; X2[31-0] := Tmp[95-64];
  - X1[31-0] := Tmp[63-32]; X0[31-0] := Tmp[31-0];
  - xmm1 := [RotWord (SubWord (X3)) XOR RCON, SubWord (X3), RotWord (SubWord (X1)) XOR RCON, SubWord (X1)]

- AESIMC xmm1, xmm2/m128
  - RoundKey := xmm2/m128;
  - xmm1 := InvMixColumns (RoundKey)

Physical Cores, Pipelining, and Hyperthreading

- Multiple physical cores in a processor greatly affect the performance of AES-NI.
- Pipelining is supported in AES-NI when using a mode of operation that can be run in parallel.
- Hyperthreading does not result in any performance gain.

Security Benefits

- Software implementations of AES are vulnerable to side channel attacks due to lookup tables being stored in cache.
  - Processes running concurrently with an encryption scheme constantly write data, forcing AES process to continuously write data.
  - Process can then determine what cache lines are being written to and eventually determine secret key.
- AES-NI does not have this vulnerability because there is no need to store lookup tables in cache.

Power Consumption

- In addition to performance gains and security gains, processes using AES-NI also benefit from lower power consumption.
  - Fewer instructions = fewer CPU cycles required
  - Everything from server farms to mobile devices benefit from less power consumption

Other Algorithms

- Other algorithms can use pieces of AES and utilize AES-NI. First round candidates from NIST hash competition examples: LANE, SHAMATA, SHAvite-3, ECHO, GrOstl, Lesamnta, and Vortex.

- Non cryptographic functions such as RAID-6 implementation.
  - Utilize AES-NI to assist in second parity block calculation.

Ease of Implementation

- Intel provides libraries for most modes of AES operation.
  - C and Assembly language examples.
  - Will run on numerous compilers to include gcc.

- Many programs support AES-NI
  - Most encryption programs (Bitlocker, TrueCrypt/VeraCrypt)
  - 7-zip, WinRAR compression software
  - OpenSSL / LibreSSL / CryptoLib packages

Advertised Performance Increases

- For non-parallel modes of AES operation (such as CBC-encrypt) AES-NI can provide 2x-3x gain in performance over a completely software approach.
- For parallelizable modes (such as CBC-decrypt, CTR, and CTR-derived modes of GCM and XTS) AES-NI can provide a 10x gain in performance over software-only solutions.

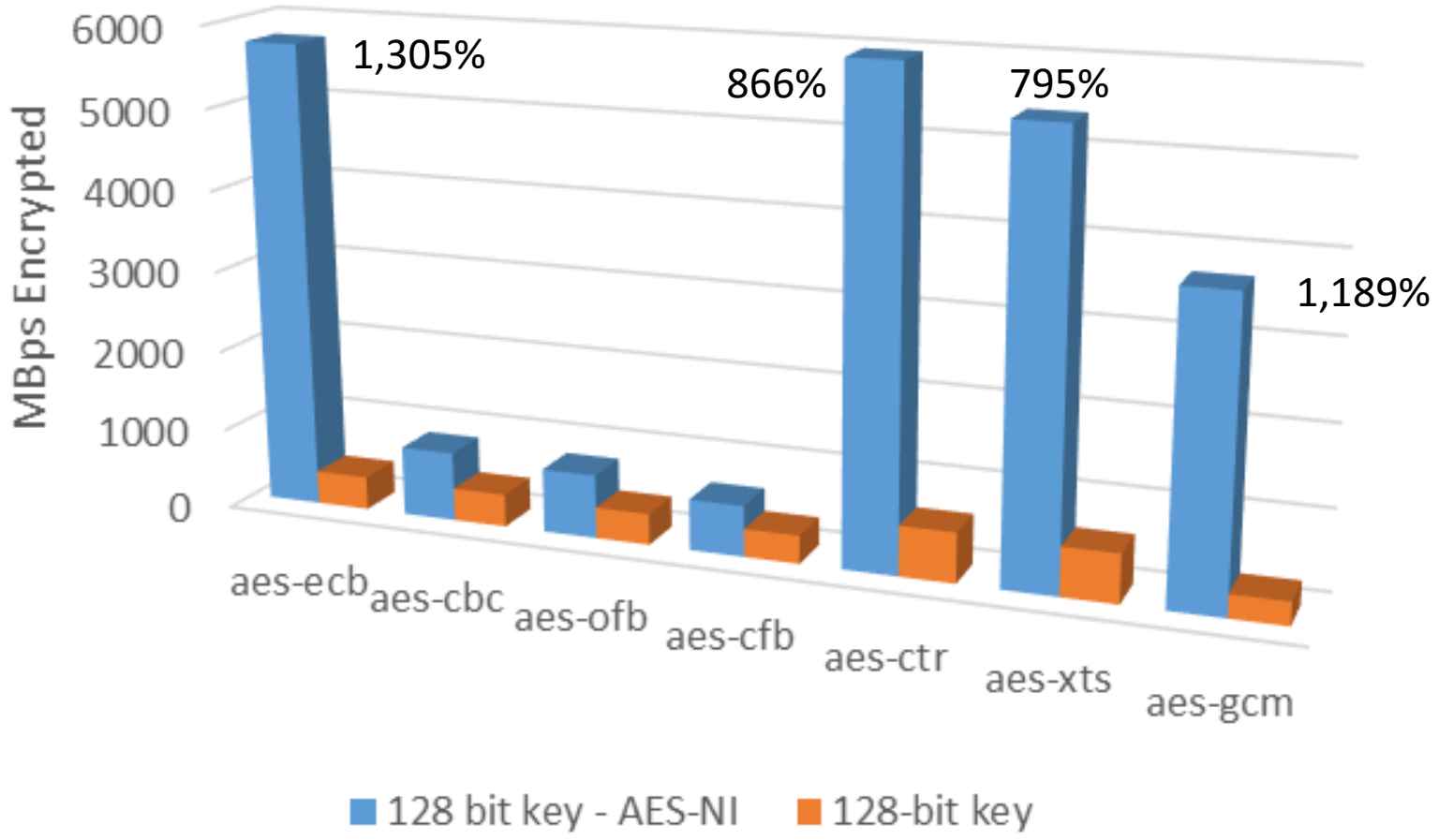
Testing Platform

- Intel Core i7-4770K CPU @ 3.50 GHz
  - Quad Core, Hyperthreading Disabled
- Intel Core i5-4670K CPU @ 3.4 GHz
  - Quad Core, Hyperthreading Disabled
- Intel Corei5-3317U CPU @ 1.7 GHz
  - Dual Core, Hyperthreading Disabled

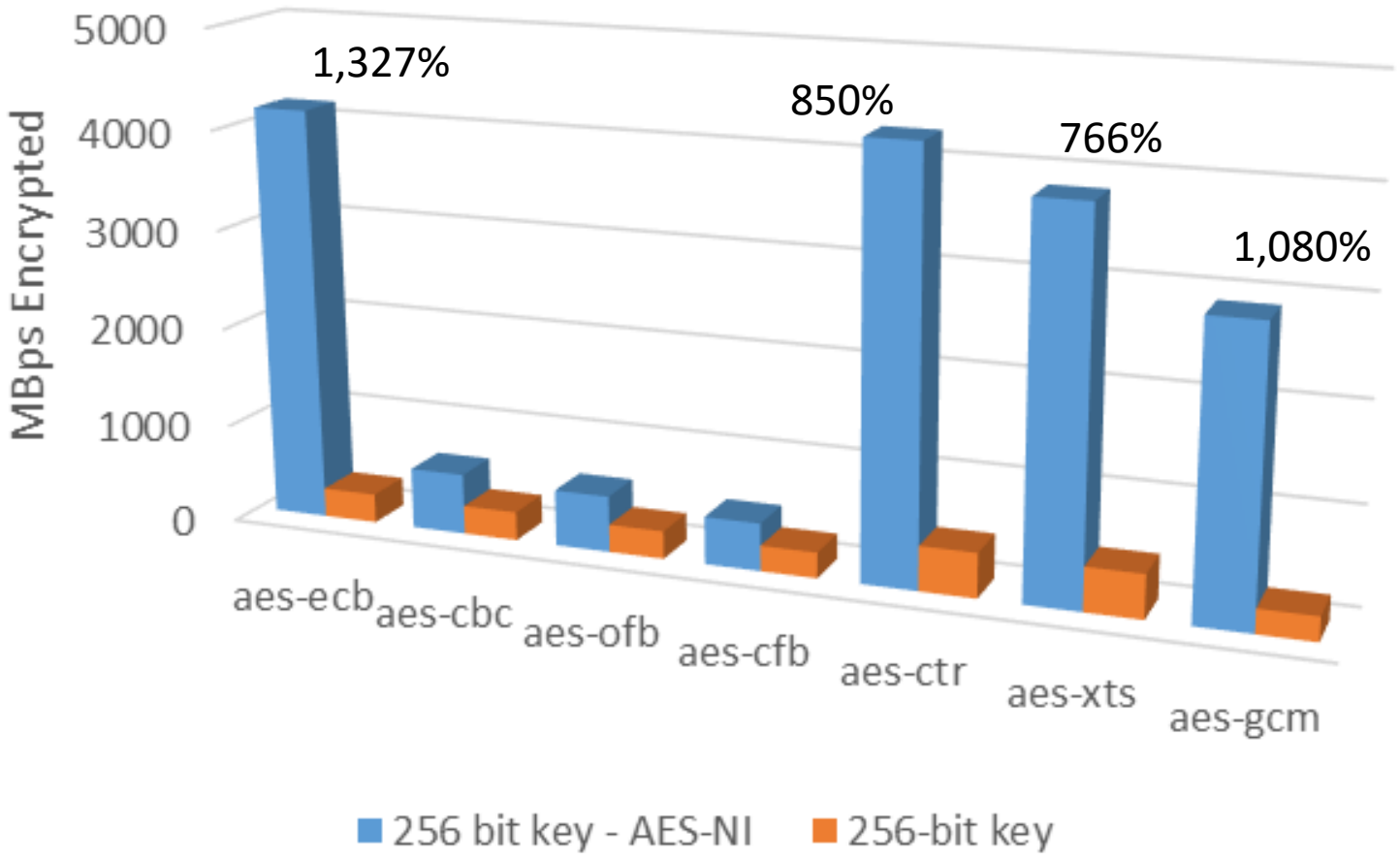
- Ubuntu 14.04 LTS
- OpenSSL 1.0.2e

Testing Methodology

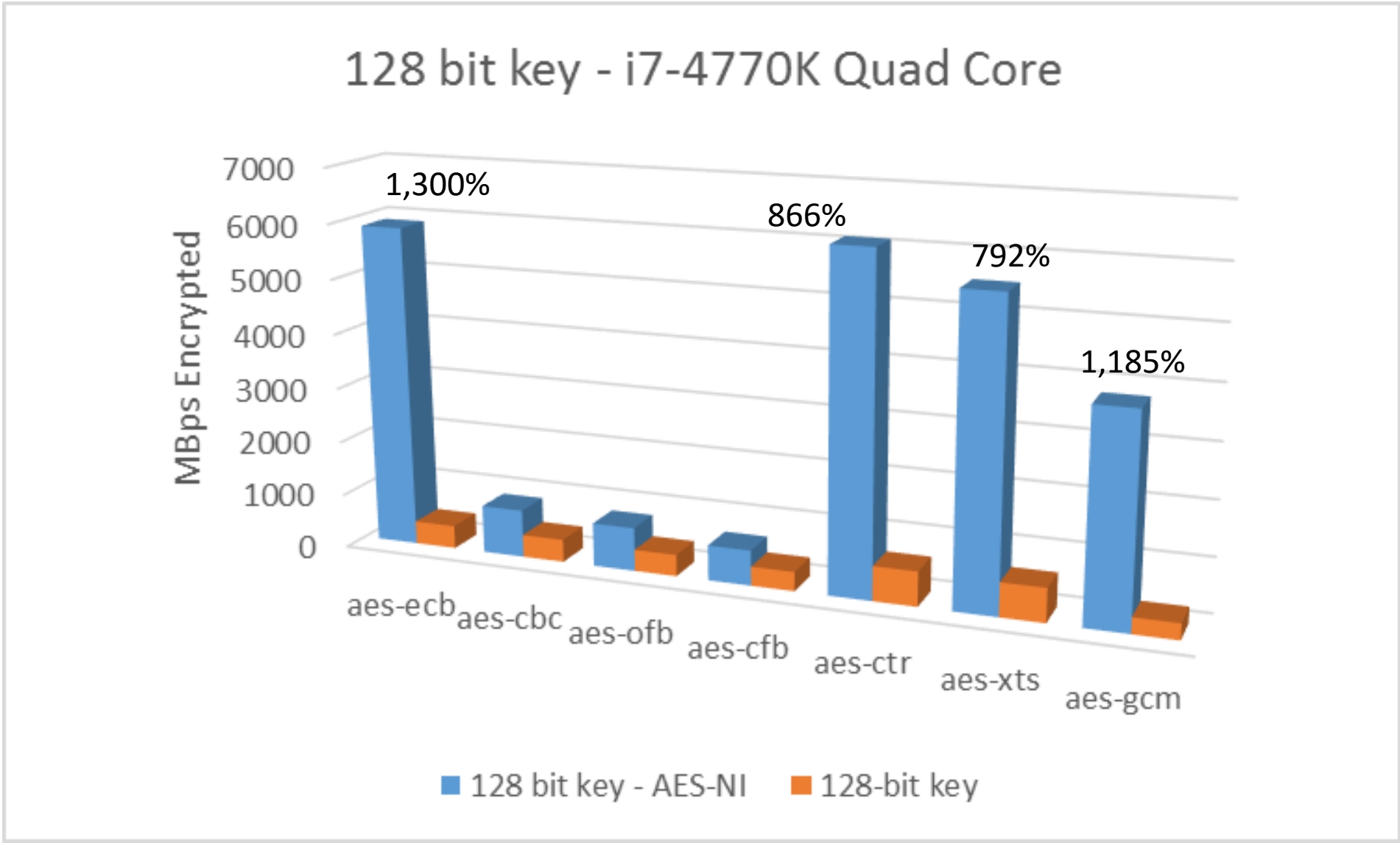
- Test seven AES modes of operations
  - ECB, CBC, OFB, CFB, CTR, XTS, and GCM
- Test all key sizes
  - This presentation limited to 128 and 256 bit keys
- Test various amounts of data to encrypt
  - This presentation limited to the encryption of 8,192 bytes.



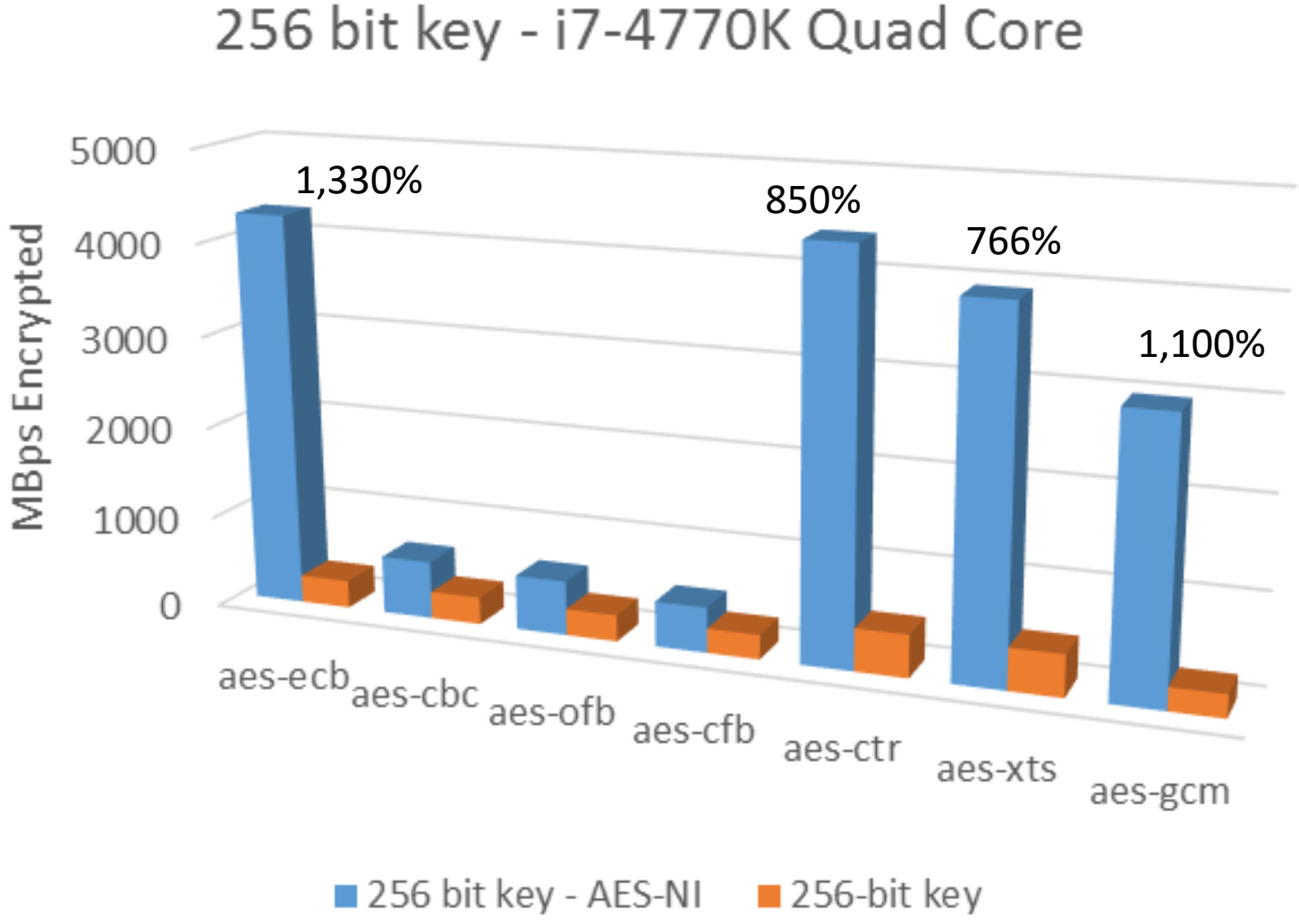
Noticeable performance increases on this testing platform. Modes of operation that can be run in parallel see greatest performance increases.



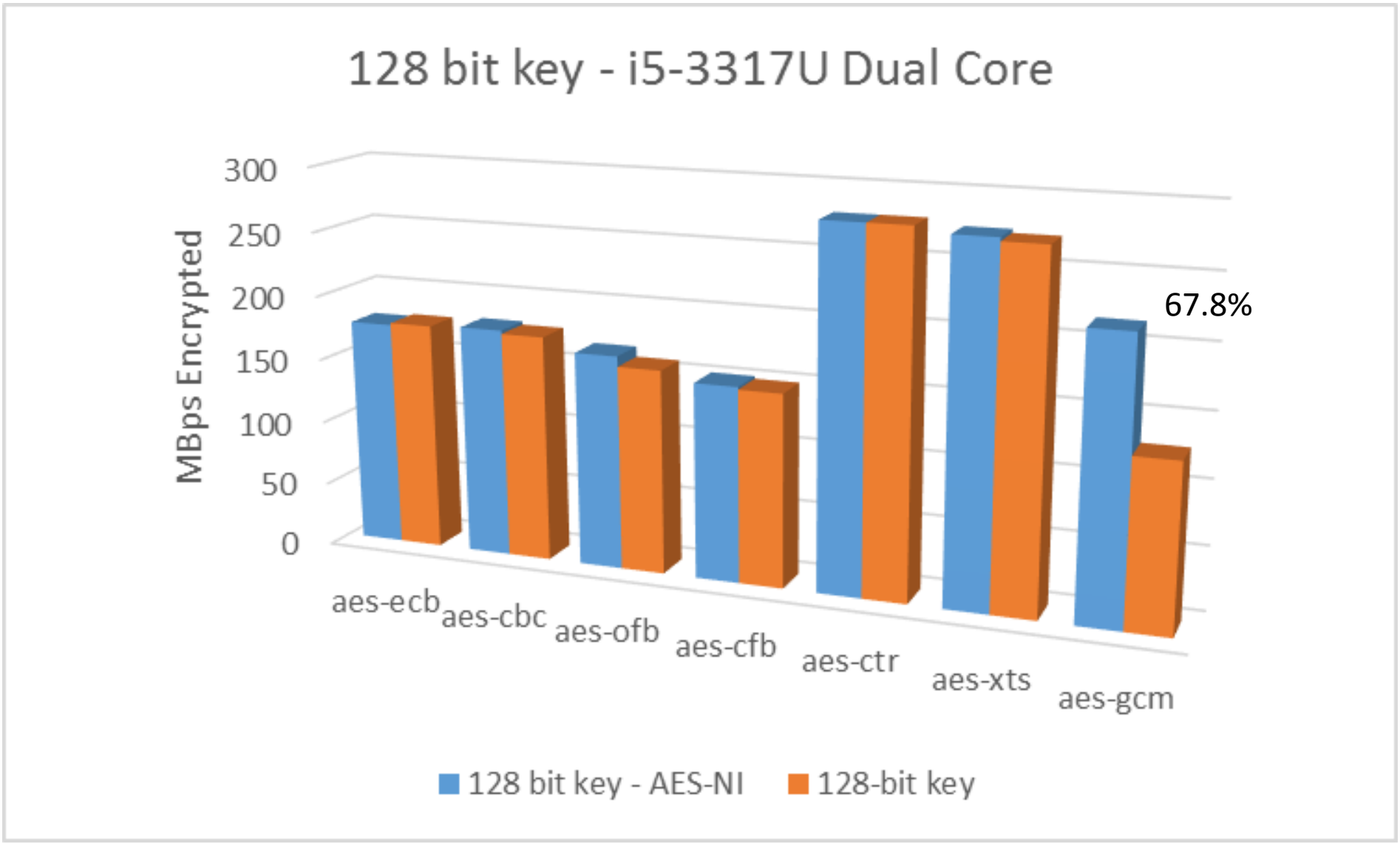
Noticeable performance increases on this testing platform. Modes of operation that can be run in parallel see greatest performance increases.



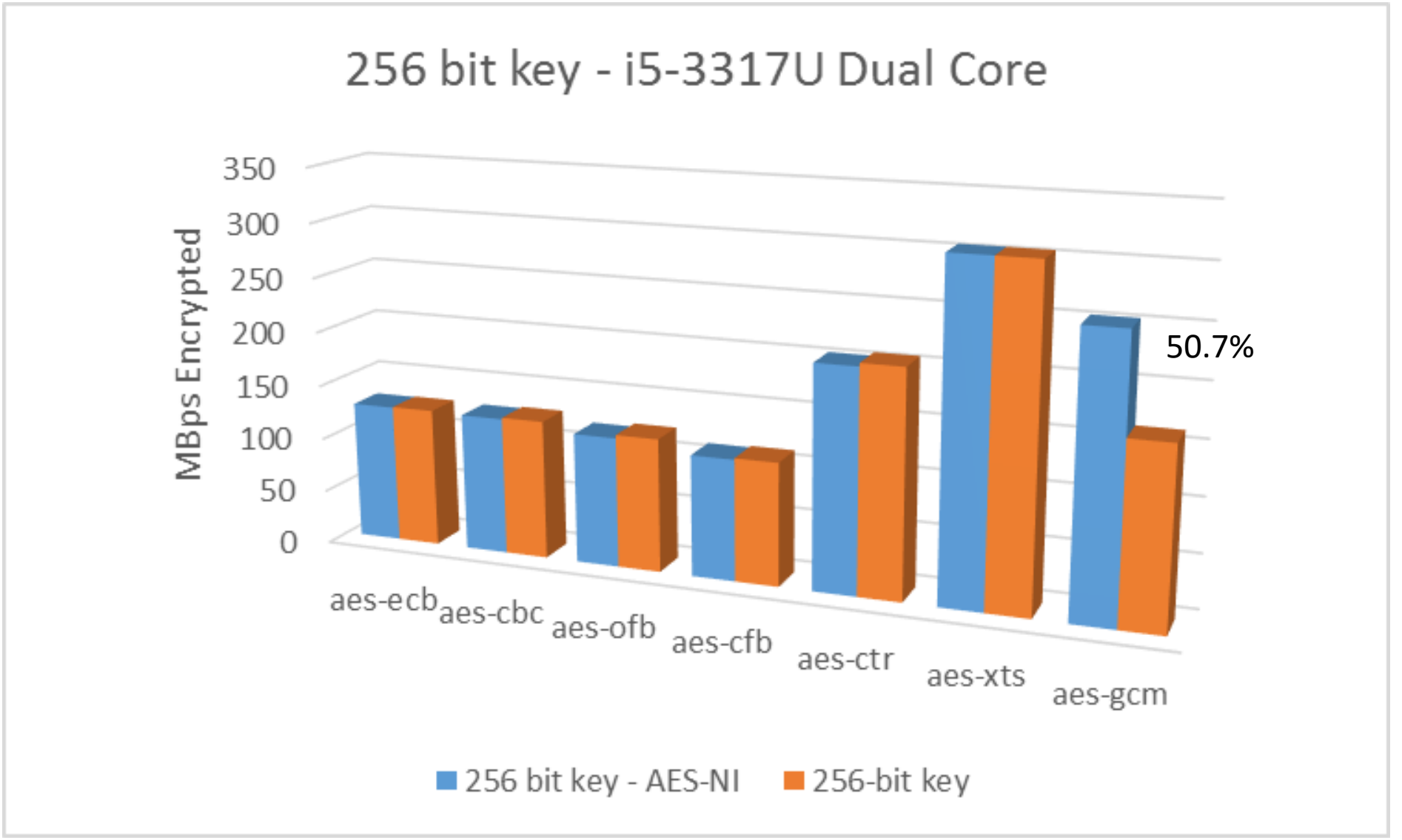
Noticeable performance increases on this testing platform. Modes of operation that can be run in parallel see greatest performance increases.



Noticeable performance increases on this testing platform. Modes of operation that can be run in parallel see greatest performance increases.



Negligible performance increases on this testing platform, with exception of gcm.



Negligible performance increases on this testing platform, with exception of gcm.

Analysis of Testing Data

- AES-NI performance was in line with or better than advertised claims from Intel.
  - Parallel modes of operation typically saw a performance increase of 7x – 13x.
  - Non-parallel modes of operation typically saw a 2x performance increase.
- # of physical cores and generation of CPU affect results.

Questions?