Intel® AES-NI Performance Testing on Linux/Java Stack

Intel Corporation

Authors

Abhi Basu
Abhilasha Bhargav-Spantzel

Contributors

Satpal Birak
Steve Deutsch
George Pappas
…

TABLE OF CONTENTS

Table of Tables

Table of Figures

1.Executive Summary

The development of sophisticated computer and networking technologies has enabled our society to be more “connected” and always “online” than ever before resulting in the generation of exponentially greater amounts of data. Such data can comprise of important information like financial, medical, personal, and even matters related to national security, so it becomes one of the most critical collaterals going forward. Therefore, it is essential to adopt advanced levels of security and privacy practices to protect our data.

In the healthcare sector, we find acts like the Health Insurance Portability and Accountability Act (HIPAA) mandating the encryption of Personal Health Information (PHI) at rest and in motion [See HIPAA Security Rule - “Implement a mechanism to encrypt and decrypt EPHI.” Rule 164.312(e)(2)(ii), 164.312(a)(2)(iv)]. However, the adoption of security technologies is sometimes bypassed due to reasons like cost, system complexity, and longer application response times (due to the additional processing time needed by the software security layer). While the increase in cost and complexity may be unavoidable, one may be able to mitigate the impact of application performance degradation through the use of innovative technologies. One example of such technology is Intel® Advanced Encryption Standards – New Instructions (AES-NI) which is hardware-based encryption/decryption that may provide enough acceleration to offset the application performance degradation due to additional security layers needed for effective data protection.

We attempted to measure the performance benefit offered by Intel® AES-NI on a Linux/Java software stack in an effort to prove that use of such technology may be beneficial for the healthcare sector and allow more organizations to address the increasing security concerns within the industry and by consumers.

By using Intel® AES-NI we were able to observe consistent and significant performance improvement in application file encryption/decryption. Specifically, **38% (average) for encryption** and **37.5% (average) for decryption**, over a wide range of key sizes and file sizes.

2.Introduction

2.1. Goal

The goal of this paper is to demonstrate performance gains obtained by using Intel® AES-NI instructions (see details in section 2.2) on a Linux/Java software stack. The first server processor series that enabled AES-NI is the Intel® Xeon® 5600 processor series.

We investigated the following two use cases with Intel® AES-NI enabled and disabled, to assess the improvements:

1. Overall Java JDK performance improvement
2. Application file encryption/decryption performance improvement

2.2.Intel® AES-NI features

Intel® AES-NI (http://www.intel.com/content/www/us/en/architecture-and-technology/advanced-encryption-standard--aes-/data-protection-aes-general-technology.html) is a set of new microprocessor instructions that implement some of the sub-steps of the AES algorithm directly in hardware, speeding up execution of the AES application and helping prevent side channel attacks. Four instructions accelerate encryption and decryption of first and last round. One instruction does the mix column operation for each round and another instruction generates the next key. CLMUL, which speeds up carry-less multiplication, is the 7th instruction in the Intel® AES-NI instruction set.

Two of the security benefits of Intel® AES-NI are broader use and resistance to side channel attacks. The built-in hardware speeds up the encryption, thereby enabling increased deployment of encryption in the data center. A broader use of encryption has the benefit of having more information protected. Pure software implementations of AES are vulnerable to side-channel attacks. Intel® AES-NI is a hardware implementation that reduces data manipulations/table lookups in caches and memory, thus lowers risk of software side-channel attacks. Hence, Intel® AES-NI enables broader use of AES and better data protection.

2.3. Objectives

This paper allows an end user of Intel® AES-NI technology to setup a benchmark mechanism on their Linux/Java software stack running on an Intel® AES-NI enabled hardware, and evaluates the benefit of leveraging the Intel® AES-NI instructions versus using a software-based Intel® AES-NI implementation. Specifically the following are the key objectives of this paper:

1. Define the operating system, software stack and tools.
2. Define the setup of Intel® AES-NI capable platform along with the software stack (operating system and API).
3. Performance testing results for data at rest:
    a. SpecJVM2008 showed 20% improvement for data at rest benchmarking.
    b. Java application showed a consistent range of improvement for various file sizes (ranging from 50MB – 1GB).
4. Security Usages and Usability
    a. Medical Device
    b. Secure Patch Management
5. Performance and Security Analysis

2.4.Audience

SW developers and technologists can use this document to understand the performance impact of Intel® AES-NI instructions in a Linux/Java software stack.

2.5.Terminology

Term	Description
AES	Advanced Encryption Standard
AES-NI	AES New Instructions
JDK	Java Development Kit
IDE	Integrated Development Environment
BKM	Best Known Method
CLMUL	Carry-less Multiplication
HIPAA	Health Insurance Portability and Accountability Act.
PHI	Personal Health Information

Table 1 - Terminology

2.6. Acknowledgements

Thanks to Aleksey Ignatenko for providing BKM to enable Intel® AES-NI for the JVM.

3. System Setup and Configuration

3.1. Components

Component	Details
Hardware	Westmere-EP Server, Xeon X5690 2-socket, 3.47 GHz with Supermicro BIOS X8DTU-LN4+ mode, 12 GB RAM, 500 GB HDD.
Operating System	Cent OS 5.6 Linux OS.
Application Software	1. Oracle JDK 1.7 2. Mozilla nss libraries 3.12.10 RTM 3. Java IDE (NetBeans 7.0) – for code development (optional)
Tools	1. CPUID 2. SPECjvm2008

Table 2 - System Components

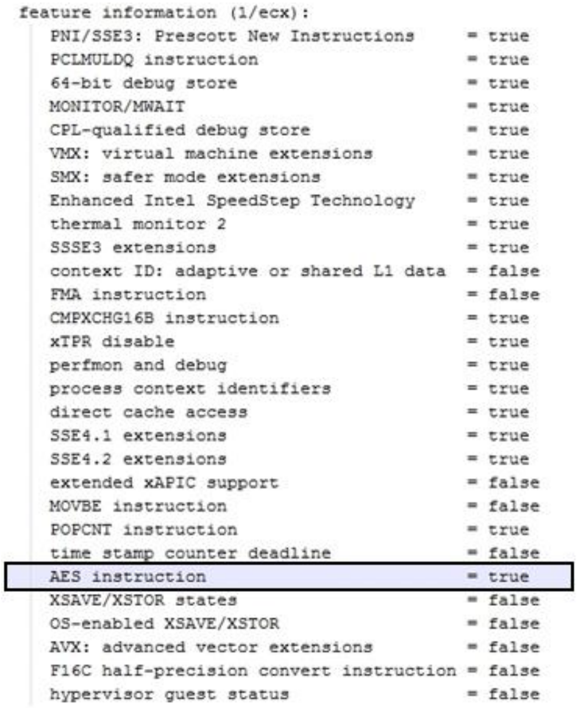
3.2. Enable/Disable Intel® AES-NI on test system

Step	How
1	Enter system BIOS by hitting F2 or Del key.
2	Go to >> Advanced >> Processor & Clock Options >> Intel® AES-NI >> select Enable or Disable.
3	Press Enter key.
4	Hit F10 to save and exit.
6	Reboot client and allow the operating system to load.

Table 3 - Enabling Intel® AES-NI in BIOS

3.3. Discover Intel® AES-NI status from local host

The Linux /proc/cpuinfo/ command does not accurately detect if Intel® AES-NI is enabled or disabled on the hardware. CPUID (http://www.etallen.com/cpuid/) tool can be used to make accurate determination. Below is an example of what CPUID output looks like within Linux, when Intel® AES-NI instructions have been enabled in the BIOS:



```
feature information (1/ecx):
   PNI/SSE3: Prescott New Instructions     = true
   PCLMULDQ instruction                    = true
   64-bit debug store                      = true
   MONITOR/MWAIT                           = true
   CPL-qualified debug store               = true
   VMX: virtual machine extensions         = true
   SMX: safer mode extensions              = true
   Enhanced Intel SpeedStep Technology     = true
   thermal monitor 2                       = true
   SSSE3 extensions                        = true
   context ID: adaptive or shared L1 data  = false
   FMA instruction                         = false
   CMPXCHG16B instruction                  = true
   xTPR disable                            = true
   perfmon and debug                       = true
   process context identifiers             = true
   direct cache access                     = true
   SSE4.1 extensions                       = true
   SSE4.2 extensions                       = true
   extended xAPIC support                  = false
   MOVBE instruction                       = false
   POPCNT instruction                      = true
   time stamp counter deadline             = false
   AES instruction                         = true
   XSAVE/XSTOR states                      = false
   OS-enabled XSAVE/XSTOR                  = false
   AVX: advanced vector extensions         = false
   F16C half-precision convert instruction = false
   hypervisor guest status                 = false
```

3.4. Software Setup

3.4.1. Enable Intel® AES-NI in Oracle JVM

The BKM to enable Intel® AES-NI support for the JVM was obtained from http://moss.amr.ith.intel.com/sites/SSG-SSD-SOTC/java-tools/Files/BKMs/NSS/nss_oracle_jdk_bkm.htm. The NSS crypto library found in the BKM allows the Oracle JVM to be enabled for using Intel® AES-NI instructions if available on the hardware.

The steps to enable AESNI support in the JVM are outlined below:

1. NSS archive can be obtained from http://ftp.mozilla.org/pub/mozilla.org/security/nss/releases/NSS_3_12_10_RTM/src/nss-3.12.10-with-nspr-4.8.8.tar.gz (AES NI is switched on if CPU supports the feature).
2. Build NSS libs.
   a. On Linux (RHEL in my case) unpack and go to mozilla/security/nss
   b. Run “make BUILD_OPT=1 USE_64=1 nss_build_all” , where USE_64=1 for x64 systems.
3. Configure JDK to use NSS crypto provider.
   a. Get all *.so files from mozilla/dist/LinuxXXXX_glibc_PTH_64_OPT.OBJ/lib
   b. Take Oracle JDK6, put NSS libs into JDK6/jre/lib/amd64
   c. Take 2 files from attach, put them in JDK6/jre/lib/security (you can see `SunPKCS11` security provider is inserted into the 1-st place there)
   d. Then run your application using the JDK you prepared

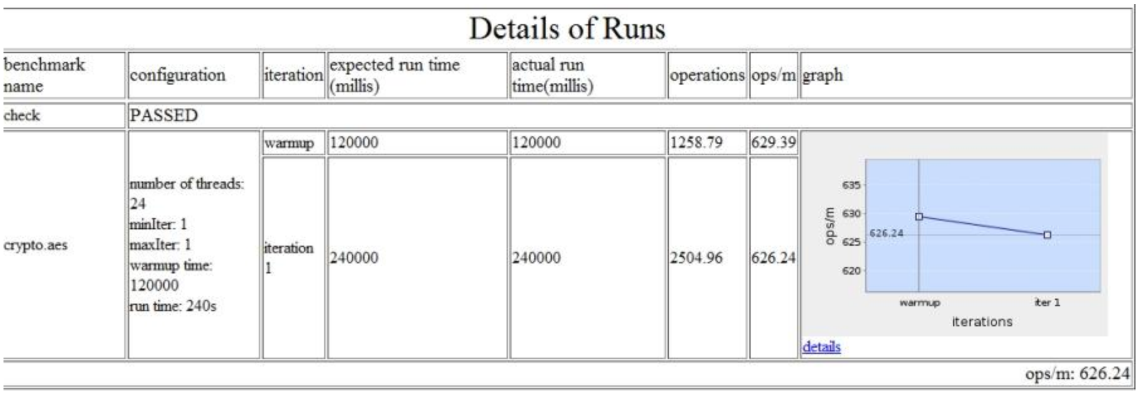
After Step 3 is complete, the system is ready for testing.

4.Test Cases

The following two types of tests were run to evaluate performance improvements:

1. **JDK performance improvement:** To test NSS libraries with Intel® AES-NI enabled, we ran crypto.aes benchmark from SPECjvm2008.
    a. Run command "java –server –jar SPECjvm2008.jar crypto.aes" from SPECjvm2008 folder.
    b. Perform tests with Intel® AES-NI ON and OFF in BIOS and see the difference.
    c. SPECjvm2008 can be downloaded from http://www.spec.org/jvm2008/ .

   Below is an example of the report provided by the SPECjvm2008 benchmarking test:

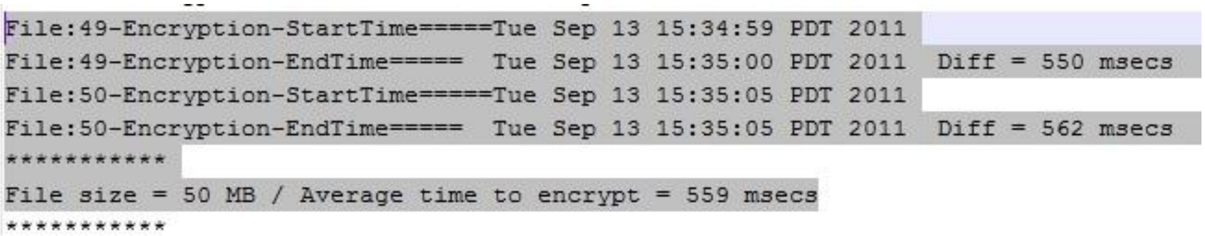


Details of Runs

benchmark name	configuration	iteration	expected run time (millis)	actual run time(millis)	operations	ops/m	graph
check	PASSED						
crypto.aes	number of threads: 24 minIter: 1 maxIter: 1 warmup time: 120000 run time: 240s	warmup	120000	120000	1258.79	629.39	details
		iteration 1	240000	240000	2504.96	626.24	

ops/m: 626.24

2. **Application performance improvement:** To test for application performance impact we used a Java application that encrypts and decrypts files of various sizes (50MB, 100MB, 200MB, 512MB and 1GB) using AES128 and AES256 keys to determine application level encryption-decryption performance. Below is an example of logs provided by the Java application:



```
File:49-Encryption-StartTime=====Tue Sep 13 15:34:59 PDT 2011
File:49-Encryption-EndTime=====  Tue Sep 13 15:35:00 PDT 2011  Diff = 550 msecs
File:50-Encryption-StartTime=====Tue Sep 13 15:35:05 PDT 2011
File:50-Encryption-EndTime=====  Tue Sep 13 15:35:05 PDT 2011  Diff = 562 msecs
***********
File size = 50 MB / Average time to encrypt = 559 msecs
***********
```

5.Performance Testing Results

5.1.JDK Performance Improvement - Operations per Minute Evaluation

SPECjvm2008 was used to compare the operations per minute (using included benchmarking tool) for the system with Intel® AES-NI instructions enabled and disabled. The results are given in Table 4.

Linux/Java Results (Cent)S 5.6 + nss libs + JDK 1.7)			
	NON AES-NI (ops/min)	**AES-NI (ops/min)**	**Acceleration (%)**
Run 1	523.36	627.18	19.84
Run 2	521.42	626.52	20.16
Run 3	523.98	626.24	19.52
Avg	522.92	626.65	**19.84**

Table 4 - AES Crypto Operations per Minute Comparison

(Higher number is better)

- ✓ *Key Finding:* ***SPECjvm2008 tool showed around 20% performance improvement when Intel® AES-NI was enabled on the JDK running on the server.***

5.2.Application Performance Improvement - Intel® AES-NI Performance Impact with File and Key Sizes

Each test run consisted of running the test 100 times (each test consisted of two test runs i.e. each test was carried out 200 times) and results were averaged.

5.2.1.AES128

		NON AES-NI		AES-NI		RESULTS	
Test Run	**Size**	**Encryption Time (msecs)**	**Decryption Time (msecs)**	**Encryption Time (msecs)**	**Decryption Time (msecs)**	**Encryption Acceleration (%)**	**Decryption Acceleration (%)**
1	50 MB	1064	1109	562	658	47.18	40.67
2	50 MB	1061	1155	575	685	45.81	40.69
					AVG	**46.49**	**40.68**
3	100 MB	2011	2113	1189	1247	40.88	40.98
4	100 MB	1996	2216	1275	1420	36.12	35.92
					AVG	**38.50**	**38.45**
5	200 MB	4303	4422	2883	2820	33.00	36.23
6	200 MB	4186	4320	2881	2783	31.18	35.58
					AVG	**32.09**	**35.90**
7	512 MB	10458	11680	6328	7421	39.49	36.46
8	512 MB	10196	11697	5855	7323	42.58	37.39
					AVG	**41.03**	**36.93**
9	1 GB	26451	24200	17235	16095	34.84	33.49
10	1 GB	26113	24111	16051	16123	38.53	33.13
					AVG	**36.69**	**33.31**

Table 5 - AES128 Encryption/Decryption Performance Numbers

(Smaller number is better)

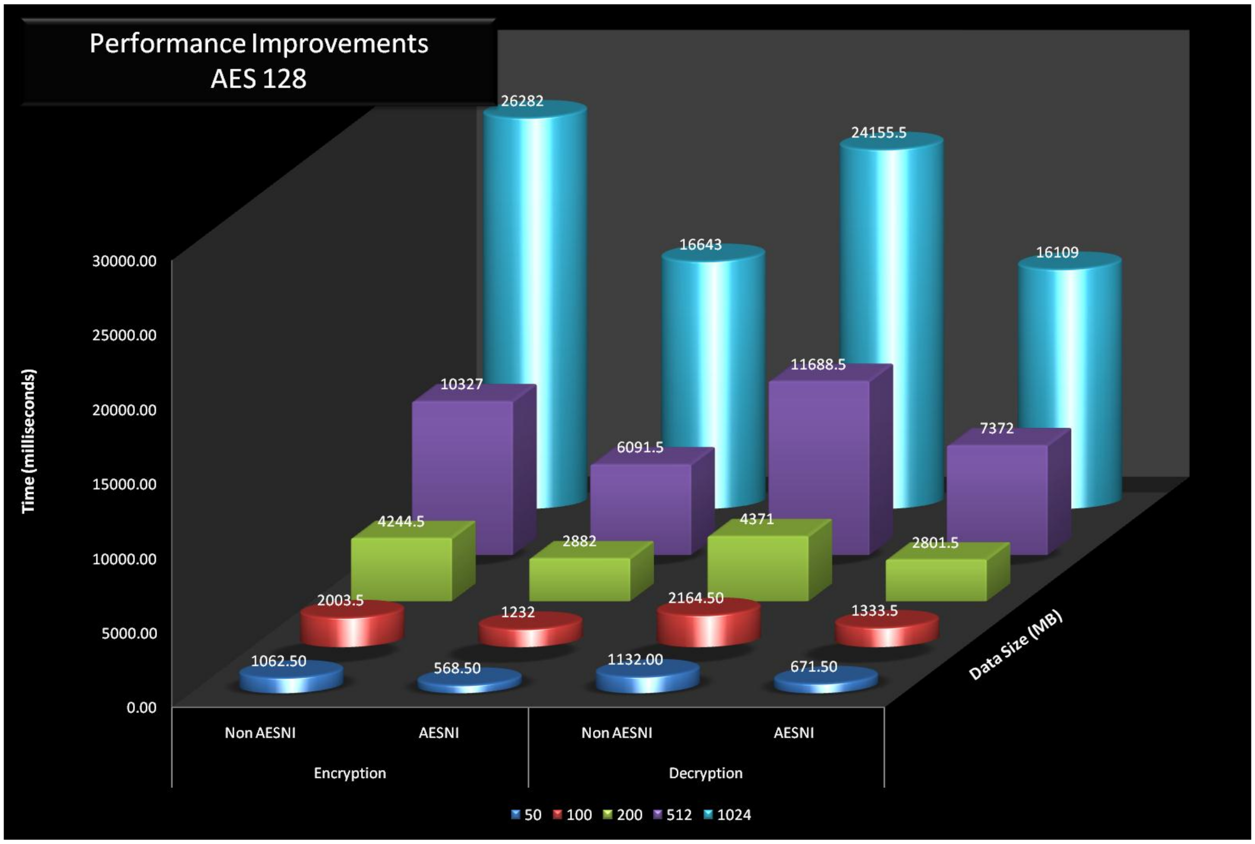


Figure 1- AES 128 Encryption/Decryption Performance Numbers

5.2.2.AES256

		NON AES-NI		AES-NI		RESULTS	
Test Run	**Size**	**Encryption Time (msecs)**	**Decryption Time (msecs)**	**Encryption Time (msecs)**	**Decryption Time (msecs)**	**Encryption Acceleration (%)**	**Decryption Acceleration (%)**
1	50 MB	1026	1066	563	677	45.13	36.49
2	50 MB	947	1084	559	638	40.97	41.14
					AVG	**43.05**	**38.82**
3	100 MB	2036	2218	1297	1388	36.30	37.42
4	100 MB	2045	2252	1274	1408	37.70	37.48
					AVG	**37.00**	**37.45**
5	200 MB	4282	4962	2984	2881	30.31	41.94
6	200 MB	4300	4989	2826	2880	34.28	42.27
					AVG	**32.30**	**42.11**
7	512 MB	10754	11729	6401	7443	40.48	36.54
8	512 MB	10081	11822	6402	7502	36.49	36.54
					AVG	**38.49**	**36.54**
9	1 GB	23976	25295	15721	16043	34.43	36.58
10	1 GB	23581	25033	15774	16266	33.11	35.02
					AVG	**33.77**	**35.80**

Table 6- AES256 Encryption/Decryption Performance Numbers

(Smaller number is better)

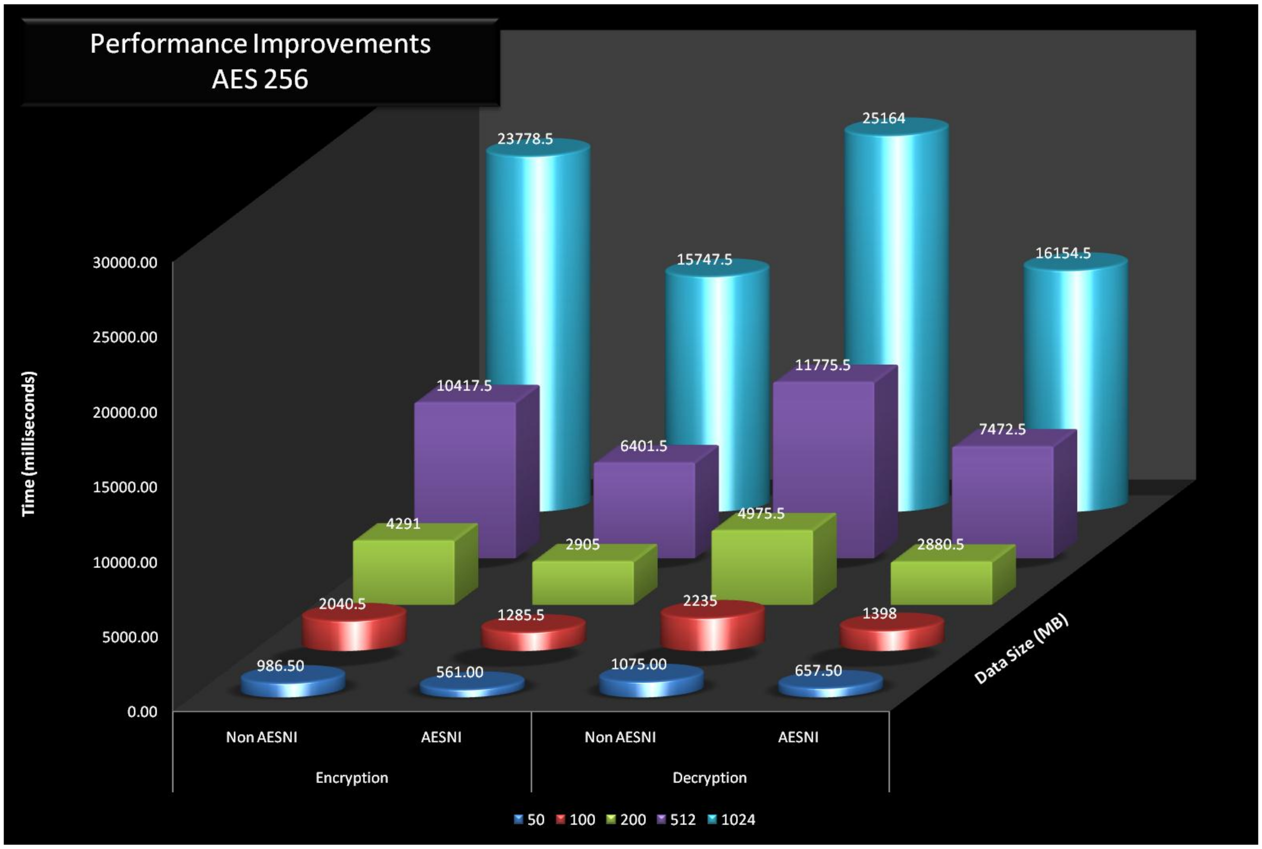
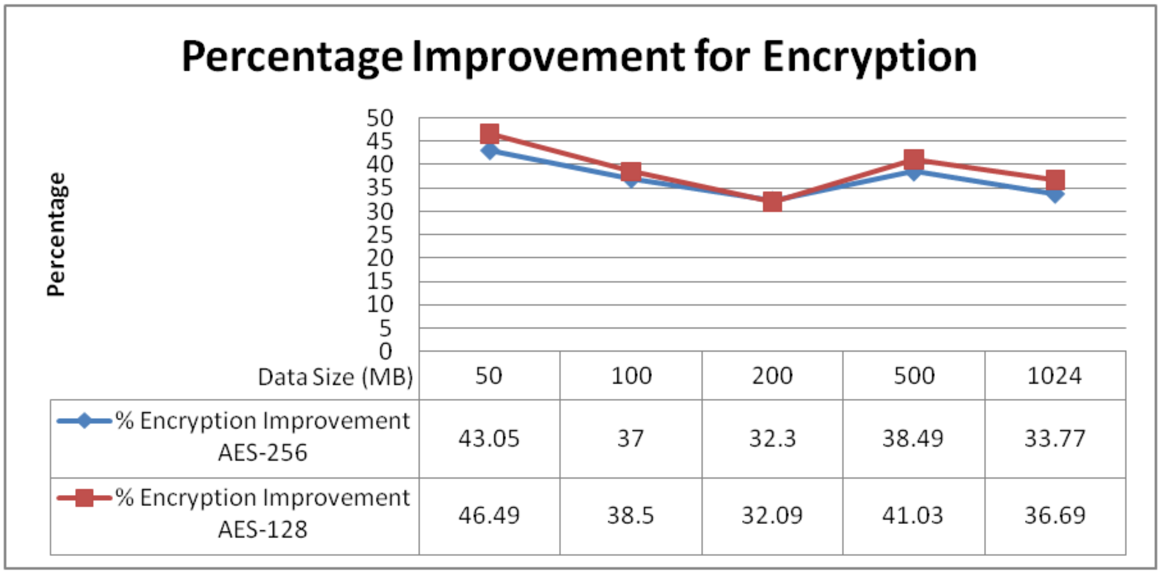


Figure 2- AES256 Encryption/Decryption Performance Numbers

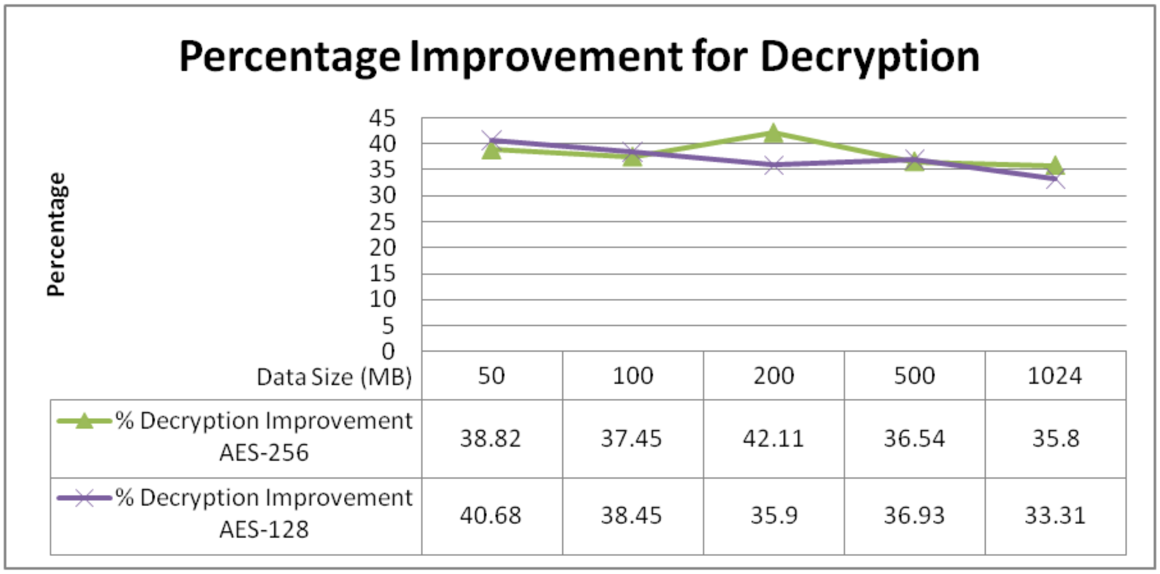
5.2.3. Percentage Improvement

The percentage improvement for encryption and decryption operations for the different key lengths and file sizes is illustrated below.



Data Size (MB)	50	100	200	500	1024
% Encryption Improvement AES-256	43.05	37	32.3	38.49	33.77
% Encryption Improvement AES-128	46.49	38.5	32.09	41.03	36.69

Figure 3 - Percentage Improvement for Encryption



Data Size (MB)	50	100	200	500	1024
% Decryption Improvement AES-256	38.82	37.45	42.11	36.54	35.8
% Decryption Improvement AES-128	40.68	38.45	35.9	36.93	33.31

Figure 4 - Percentage Improvement for Decryption

- ✓ **Key Findings**
  - ***Application file encryption improved 39% (average) and file decryption 37% (average) with Intel® AES-NI enabled over AES128 key.***

  - ***Application file encryption improved 37% (average) and file decryption 38% (average) with Intel® AES-NI enabled over AES256 key.***

Note:

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design, configuration or workload may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, visit Intel Performance Benchmark Limitations.

Our next steps would be to try to replicate these steps on a Windows/Java stack to see if we can observe similar or better trends.

6.Security Usages and Usability Discussion

The widespread use of the Internet, more connected models, and sophisticated computer and networking technologies have created the most important and critical collateral – "data". Currently, a big percentage of data or information exchanged over the Internet today is not secure or encrypted effectively. In the healthcare arena, data security becomes crucial as individual PHI can be compromised to criminals who can use that valuable information to wreak havoc to interconnected systems and personal lives of innocent citizens. Examples of some commonly used healthcare workflows are data storage in EMR (Electronic Medical Records), secure messaging, physician-patient communication, lab results transferred to EMR, Patient Portals and ePrescibing.

Although data encryption is becoming mandatory to meet organizational confidentiality requirements, there are various reasons why it is sometimes circumvented. The top reasons are expense, system or software complexity and longer application response times (due to the need for encryption and decryption layer). Many hospitals, clinics and health care facilities are transferring un-encrypted data between systems due to one of the reasons mentioned above. Technologies like Intel® AES-NI allow better performance of systems by providing seamless encryption/decryption in the hardware layer as opposed to the higher software layer like software-based encryption systems, and try to alleviate the usage aspect complains of data encryption.

Below is an example of some typical file sizes and what form of information they may represent to provide some context to this experiment.

Size	Type	Example of use case
50 MB	Patch Management	Virus patch update
100 MB	Data Transfer / Medical Device	PHI data download from Central Server to Healthcare worker's laptop.
200 MB	Data Transfer	Lab results transferred to EMR
512 MB	Medical Device	Medium-quality X-Ray image
1 GB	Medical Device	High-quality 3D X-Ray image

Table 7 - Examples of Typical File Sizes

Based on the use of a given type of file one can retrieve the acceptable response or processing time of the data. One way to highlight the benefit of Intel® AES-NI technology would be to demonstrate that Intel® AES-NI can be used to accelerate the customer's application response times and improve security without impacting usability.

7.Conclusion and Future Work

The Linux/Java stack in this test environment showed consistent and considerable performance gains for encryption/decryption using Intel® AES-NI enabled hardware.

In the medical field, where privacy and security of data is crucial, Intel® AES-NI can provide the much needed performance improvement to offset the overhead of encryption/decryption like higher data access times. This becomes more significant when utilizing Full Disk Encryption (FDE) where the entire system including the operating system is encrypted, and has to work transparently to the end user. We plan to extend this work to incorporate performance measurement for FDE applications.

8.Further Reading

- www.intel.com/technology/dataprotection
- Shay Gueron, “Advanced Encryption Standard (AES) Instruction Set rev 2”, Intel whitepaper, June 2009. http://software.intel.com/en-us/articles/advanced-encryption-standard-aes-instructions-set/
- Shay Gueron, Michael Kounavis, “carry-less multiplication and its usage for computing the GCM Mode”, Intel whitepaper, August 2009. http://software.intel.com/en-us/articles/carry-less-multiplication-and-its-usage-for-computing-the-gcm-mode/PCLMULQDQ
- http://www.cs.bc.edu/~straubin/cs381-05/blockciphers/rijndael_ingles2004.swf
- IDF Sessions – September 22-24, 2009 www.intel.com/idf/sf09/audio_sessions.htm
    - ➢ **ECTS003** – Intel(R) AES-NI: New Technology for Improving Encryption Efficiency and Enhancing Data Security in the Enterprise Cloud Securing the Enterprise with Intel(R) Intel® AES-NI
    - ➢ **ECTS002** - Intel® Trusted Execution Technology: A More Secure Launch Environment for the Enterprise Cloud
- 2010 IDF Session https://intel.wingateweb.com/us10/scheduler/catalog/catalog.jsp
    - ➢ **DCCS002** - Securing Today’s Data Centers Against Tomorrow’s Attacks