Introduction to Intel® AES-NI and Intel® Secure Key Instructions

Gael Hofemeier and Robert Chesebrough

Table of Contents

Introduction

Starting in 2010 with the Intel® Core™ processor family based on the 32nm Intel® microarchitecture, Intel introduced a set of new AES (Advanced Encryption Standard) instructions. This processor launch brought seven new instructions. As security is a crucial part of our computing lives, Intel has continued this trend and in 2012 and has launched the 3rd Generation Intel® Core® Processors, codenamed Ivy Bridge. Moving forward, 2014 **Intel® microarchitecture code name Broadwell** will support the RDSEED instruction.

The following paragraphs provide a high level overview of the Intel® AES-NI extensions and the Intel® Secure Key Instructions (RDRAND and RDSEED.) This paper will not go into deep technical detail, but will provide a description, where to go for code samples and help, and where to find out more information about the instructions.

Terminology

AES	Advanced Encryption Standard
AES-NI	AES New Instructions
DRBG	Deterministic Random Bit Generator
RNG	Random Number Generator
DRNG	Digital Random Number Generator
GCM	Galois Counter Mode
NRBG	Non-Deterministic Random Bit Generator

Intel® AES-NI

While there are technically six AES instructions supported on selected Intel processors, we often see an additional instruction documented along with them. This additional instruction is the PCLMULQDQ instruction and its significance will be explained here as well.

The AES-NI extensions offer full hardware support for data encryption and decryption using the Advanced Encryption Standard, defined by FIPS Publication number 197. Four of the instructions support AES Encryption and Decryption while the other two support AES key expansion.

The AES-NI extensions have the flexibility to support key lengths of 128, 192, and 256 by processing the data block in 10, 12, and 14 rounds of cryptographic transformations. Since they are hardware-based, they also offer a significant increase in performance compared to the current software implementations.

Beyond improving performance, the AES instructions provide important security benefits. By running in data-independent time and not using tables, they help in eliminating the major timing and cache-based attacks that threaten table-based software implementations of AES. In addition, these instructions make AES simple to implement, with reduced code size, which helps reducing the risk of inadvertent introduction of security flaws, such as difficult-to-detect side channel leaks.

Use the RDRAND instruction to make high-quality, robust keys for AES-NI use.

Instructions/Descriptions

The AES-NI extension provides six instructions to accelerate symmetric block encryption/decryption of 128-bit data blocks using the Advanced Encryption Standard (AES) specified by the NIST publication FIPS 197.

- AESDEC and AESDECLAST – targets the AES decryption rounds using the Equivalent Inverse Cipher.
- AESENC and AESENCLAST - targets the AES encryption rounds.
- AESIMC - targets the Inverse MixColumn transformation primitive.
- AESKEYGEN - targets generation of round keys from the cipher key for the AES encryption/decryption rounds.
- PCLMULQDQ – Performs carry-less multiplication of two 64-bit data into a 128-bit result. Carry-less multiplication of two 128-bit data into a 256-bit result can use PCLMULQDQ as building blocks. Carry-less multiplication is an important piece of implementing Galois Counter Mode (GCM) operation of block ciphers. GCM operation can be used in conjunction with AES algorithms to add an authentication capability. GCM usage models also include IPsec, storage standard, and security protocols over fiber channel. Additionally, PCLMULQDQ can be used in calculations of hash functions and CRC using arbitrary polynomials.

Keep in mind that only select Intel processors have support for the AES-NI instructions. Logic should be added to all code that checks for processor support for both the AES-NI instructions, as well as for the PCLMULQDQ instruction (See Section 11.6.2: Checking for SSE/SSE2 Support in the Software Developer's Manual listed below):

- Check that the processor supports AESNI: CPUID.01H:ECX.AESNI[bit 25] = 1
- Check that the processor supports PCLMULQDQ: CPUID.01H:ECX.PCLMULQDQ[bit 1] = 1

For a complete description of all of the AES-NI extensions as well as the PCLMULQDQ instruction, please refer to section 12.13 in the Intel® 64 and IA-32 Architectures Software Developer's Manual Combined Volumes: 1, 2A, 2B, 2C, 3A, 3B and 3C

Getting Started

There are many resources on the Intel Software Network that are designed to educate developers about the AES-NI extensions. Also provided is sample code to help developers get started in using the instructions. Links for these resources are proved below under Resources and Sample Code. Should the reader have further questions about any of the instructions, questions can be posted in the discussion forum, link also provided in item number 9.

Resources and Sample Code

1. Intel® 64 and IA-32 Architectures Software Developer's Manual Combined Volumes: 1, 2A, 2B, 2C, 3A, 3B and 3C
2. Intel® Advanced Encryption Standard (AES) Instructions Set - Rev 3
3. Intel® Advanced Encryption Standard Instructions (AES-NI)
4. FIPS Publication number 197
5. AES-NI Code Sample
6. Improved Advanced Encryption Standard (AES) Crypto performance on Java with NSS using Intel® AES-NI Instructions
7. Intel® AES-NI Performance Testing on Linux/Java Stack
8. Developer Opportunities for Business PC Security
9. vPro Developers Community and Discussion Forum

Intel® Secure Key Instructions

Intel® Secure Key is currently Intel's branding for the Intel® 64 Architecture instruction RDRAND and its underlying Digital Random Number Generator (DRNG) hardware implementation. Intel Secure Key is available on the 3rd Generation Intel® Core™ processors launched in early 2012. The RDSEED instruction is part of the Intel® Secure Key family of instructions and will become available with Intel® processors codenamed Broadwell.

DRNG is a key enabler for Information Security Applications. Cryptographic protocols rely on RNGs for generating keys and fresh session values (e.g., a nonce) to prevent replay attacks. In fact, a cryptographic protocol may be algorithmically robust but vulnerable to attack due to the available random numbers being predictable or of poor quality. The Digital Random Number Generator can be used to fix this weakness, thus significantly increasing cryptographic robustness.

Closely related are government and industry applications. Due to information sensitivity, many such applications must demonstrate their compliance with security standards like FISMA, HIPPA, PCIAA, etc. RDRAND has been engineered to meet existing security standards: NIST SP800-90A, FIPS 140-2, and ANSI X9.82, and thus provides an underlying RNG solution that can be leveraged in demonstrating compliance with information security standards.

Other uses of Digital Random Number Generation include:

- Communication protocols,
- Monte Carlo simulations and scientific computing,
- Gaming applications,
- Bulk entropy applications like secure disk wiping or document shredding, and
- Protecting online services against RNG attacks.

Instructions/ Description

Currently within the Intel® Secure Key Technology, RDRAND is available on selected **Intel® microarchitecture code name Ivy Bridge platforms** launched in 2012) and RDSEED which will be available starting with Broadwell platforms. The following paragraphs provide descriptions for each instruction.

RDRAND

RDRAND returns random numbers that are supplied by a cryptographically secure, Deterministic Random Bit Generator (DRBG). All Intel processors that support the RDRAND instruction indicate the availability of the RDRAND instruction via reporting

CPUID.01H:ECX.RDRAND[bit 30] = 1

The DRBG is designed to meet the NIST SP 800-90A standard. The DRBG is re-seeded frequently from a on-chip non-deterministic entropy source to guarantee data returned by RDRAND is statistically uniform, non-periodic and non-deterministic.

For a complete description of all of the RDRAND instruction, please refer to section 7.3.17 in the Intel® 64 and IA-32 Architectures Software Developer's Manual Combined Volumes: 1, 2A, 2B, 2C, 3A, 3B and 3C

RDSEED

RDSEED returns random numbers that are supplied by a cryptographically secure, enhanced Non-Deterministic Random Bit Generator (Enhanced NRBG). All Intel processors that support the RDSEED instruction will indicate the availability of the RDSEED instruction via reporting

CPUID.(EAX=07H,ECX=0H):EBX.RDSEED[bit 18] = 1

The NRBG is designed to meet the NIST SP 800-90B and NIST SP800-90C standards. The RDSEED instruction will be a companion to the RDRAND instruction, completing Intel's solution to the "platform entropy problem" and will become available on the Intel architecture code name Broadwell client/server processors. The RDSEED instruction will provide a high quality/high performance Entropy Source for (re)seeding software Pseudo Random Number Generators (PRNGs), for example, Microsoft*, Linux*, OpenSSL, and RSA. As with RDRAND, RDSEED will avoid any OS or Library enabling dependencies and can be used directly by any software at any protection level or processor state. RDSEED uses a small extension to the existing underlying DRNG that is used to support RDRAND. RDSEED (along with RDRAND) fills out Intel's standards compliant (e.g., NIST SP800-90 A, B, and C) HW-based Random Number Generator portfolio.

Getting Started

There are many resources on the Intel Software Network that are designed to educate developers about Intel® Secure Key. Also provided is sample code to help developers get started in using the instructions. Links for these resources are proved below under Resources and Sample Code. Should further questions arise, they can be entered in the discussion forum listed in item 7, below. Note that because RDSEED will not be available in hardware until the Intel architecture code name Broadwell timeframe, resources are limited.

RDRAND Resources and Sample Code

1. Intel® 64 and IA-32 Architectures Software Developer's Manual Combined Volumes: 1, 2A, 2B, 2C, 3A, 3B and 3C
2. Intel® Secure Key Software Implementation Guide
3. Intel® Secure Key Code Samples
4. Behind Intel's New Random-Number GeneratorNIST SP 800-90A
5. FIPS 140-2
6. vPro Developers Community and Discussion Forum

RDSEED Resources

1. TBD

Instruction Support in 3rd Party Libraries

One question that a software developer may have is whether or not a third-party software library supports Intel® AES-NI or RDRAND instructions. If the software's support website does not offer this information, there is a tool that can be used to "look" at third party libraries and determine if any of the AES-NI or RDRAND instructions are present. In order to analyze an application, 3rd party library, DLL, Linux* or shared object makes use of the above instructions. Of course one could search the 3rd party library creator's website or a forum to learn what versions of a library included AES-NI or RDRAND, but sometimes it is handy to have direct confirmation that the instructions have been built in. This is a simple matter to do on most systems, either Linux* or Windows*.

On Linux:

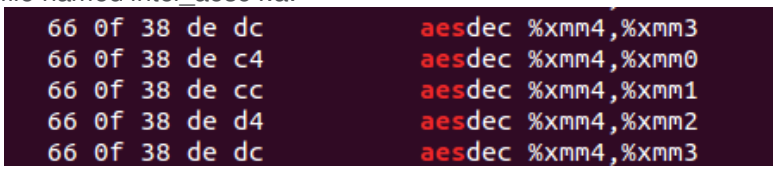
The "objdump –D" command can be used and the output piped to grep on a library or shared object (x.a or x.so) to see if the any of the six AES-NI or RDRAND instruction are included in the binary.
Here is a simple example for the AES-NI instructions that can be downloaded: AES-NI Sample Library code Complete the build steps, and issuing the following commands:

```
%objdump -D intel_aes64.a | grep aesenc
%objdump -D intel_aes64.a | grep aesenclast
%objdump -D intel_aes64.a | grep aesdec
%objdump -D intel_aes64.a | grep aesdeclast
%objdump -D intel_aes64.a | grep aeskeygenassist
%objdump -D intel_aes64.a | grep aesimc
```

A short cut is to abbreviate the command by simply looking for *aes*, but this test can lead to false positive results if subroutines or functions are given a name including aes.

Below is an example of using objdump to examine the contents of the intel_aes64.a library from the Intel AES-NI Sample Library code.

The figure below show the output of such a disassembly search on the Linux targeted Intel_aes_library file named intel_aes64.a:



This method works on Linux-based library files (with .a extension), shared object files (with .so extensions), or binary executable file (such as a.out)

The reader can try another simple example by downloading RDRAND Library, completing the build steps, and issuing the following commands:

```
%objdump -D test_rdrand | grep rdrand
```

On Windows*:

It would seem to be just as easy a matter on Windows to dump a disassembly listing of a windows binary by using DUMPBIN.exe from the MSVS* bin folder, which is built on top of the MS linker. However, while DUMPBIN appears to work great for disassembly of most x86 instructions, DUMPBIN currently does not support disassembling the AES-NI instructions or RDRAND. However, with the use of a nice cross platform binary converter and disassembler we can get the job done easily.

The following analysis was done using an Intel® Core™ i7 platform running Windows* 7. The Intel AES-NI Sample Library code was downloaded, unzipped and built with MSVS* command line and the mk_win64_all.bat command that comes with the library.

The cross platform binary conversion application/disassembler was downloaded and executed on the Windows* 7 platform. The disassembler, from www.agner.org, called objconv.exe was used to directly disassemble a Windows* dll, and the same tool was used to extract object files from a library file and then directly disassemble the extracted object files in order to find AES-NI instructions embedded in the libraries.

Below is the location to download objconv.exe from www.agner.org (Download, Manual)

The objconv.exe is run from the command line on Windows*. It is handy to set the path in the MSVS* command prompt to include the path to the folder containing objconv.exe, which in this case was C:\Users\racheseb\Desktop\Dumbin\ObjConv. Your file location will be different, of course. See figure below.

path = %path%; C:\Users\racheseb\Desktop\Dumbin\ObjConv

This next code example comes from the free Crypto++® Library 5.6.1 at www.cryptop.com. Download the zip file, and use MSVS to open the sln file, and rebuild the solution.. The reader is advised to search for the output folder that contains the DLL and Lib files and navigate to this folder from within the command window.

Dump the content of a Windows DLL or EXE to assembly:
objconv -fasm cryptopp.dll cryptopp.asm:

c:/> type cryptopp.asm | grep aes

Extract the object files from the library:

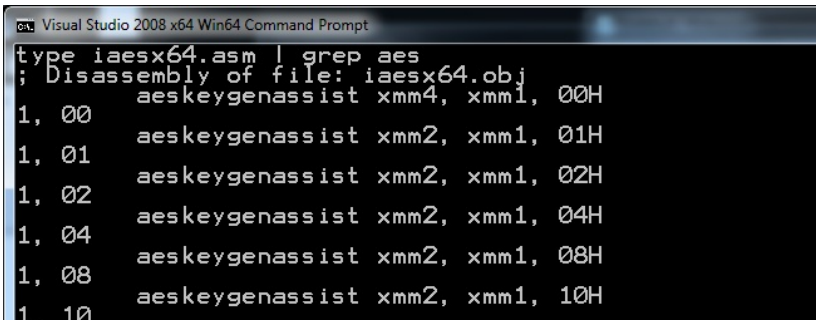
c:/> objconv -fpecoff -lx: intel_aes64.lib

Dump the content of one of these object files to assembly (requires first disassembling the suspect obj to masm by executing the following command:

c:/> objconv -fmasm iaesx64.obj iaesx64.asm

Display the result of the disassembly, showing only the aes instructions grep as follows:
c:/> type cryptopp.asm | grep aes

See output below:



Using the above steps, it is easy to search both Windows* and Linux* static and dynamic libraries, object files, and executable binaries for the presence of Intel AES-NI and RDRAND instructions.

Conclusion

Computer security is based on the use of cryptography. Robust cryptography is based on two things: Good algorithms (e.g., AES) and High quality keys (e.g., good random numbers). With AES-NI and the RDRAND instructions, Intel is making the long standing "Platform Entropy Problem" "go away" on future Intel-based platforms.

Intel continues to provide leadership in developing instruction-set extensions with support for Advanced Encryption Standard (AES) as well as the Intel Secure Key instructions. Software developers have the option of accessing all of these security instructions themselves by using 3rd party libraries that support the instructions, or they can access the instructions directly in their software. For developers who are writing their own implementations, Intel provides sample code for both the AES-NI set of instructions as well as for the RDRAND instruction. This paper also presented a tool that can be used for running against a 3rd party library in order to determine if any of the instructions are supported (in the case that the product web page or documentation does not indicate support.)

Notices

conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to:
http://www.intel.com/design/literature.htm

+Intel® AES-NI requires a computer system with an AES-NI enabled processor, as well as non-Intel software to execute the instructions in the correct sequence. AES-NI is available on select Intel® processors. For availability, consult your reseller or system manufacturer.

+Intel® Secure Key Technology requires a computer system with an Intel® 3rd Generation Core i5 or i7 processor, as well as non-Intel software to execute the instructions in the correct sequence. Intel Secure Key is available on select Intel® processors. For availability, consult your reseller or system manufacturer. For more information, see http://software.intel.com/en-us/articles/intel-digital-random-number-generator-drng-software-implementation-guide/

++Results have been estimated based on internal Intel analysis and are provided for informational purposes only. Any difference in system hardware or software design or configuration may affect actual performance.

++Performance Results have been estimated based on internal Intel analysis and are provided for informational purposes only. Any difference in system hardware or software design or configuration may affect actual performance.

Optimization Notice

http://software.intel.com/en-us/articles/optimization-notice/

Performance Notice

For more complete information about performance and benchmark results, visit www.intel.com/benchmarks

Optimization Notice (updated March 2012) or see short form above

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel.

Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

ABOUT THE AUTHORS

Robert Chesebrough is a technical analyst in the Partner Experience team with Developer Relations Division, and is responsible for bringing new security related content to the Intel software developer community. He holds a BS in Physics from the University of New Mexico and has been a software developer for various national laboratories since the early 1980's and also in the aerospace industry at SBS technologies in the late 1990's. He is married and has two children.

Gael is a Software Engineer in the Developer Relations Division at Intel working with Business Client Technologies. Gael holds a BS in Math and an MBA, both from the University of New Mexico. Gael enjoys hiking, biking, photography and is pondering having chickens as pets.