Journal of Basic and Applied Engineering Research
Print ISSN: 2350-0077; Online ISSN: 2350-0255; Volume 1, Number 6; October, 2014 pp. 90-93
© Krishi Sanskriti Publications
http://www.krishisanskriti.org/jbaer.html

Instruction Set Extensions for Cryptographic Applications: Survey and Issues

Mahendra Pratap Singh[1], Manoj Kumar Jain[2]

[1]*Navrachana University, Vadodara, India*
[2]*Computer Science, Mohanlal Sukhadia University, Udaipur, India*

Abstract: **Security in data communication is of utmost importance and is achieved using cryptographic algorithms while preserving confidentiality and integrity of data. In processors those implement cryptographic algorithms; critical computations are migrated from processor core to an application specific unit for performing computing-intensive task efficiently where each unit executes application specific instruction set extensions. Instruction set extension is one of the ways to improve the performance of processors running cryptographic applications. It is an effective way to execute computing intensive task of encryption and decryption. There is a significant performance improvement using instruction set extensions like Intel's AES-NI (Advanced Encryption Standard New Instructions) as compared to other software implementations. This paper is an attempt to survey the available instruction set extensions used in cryptographic applications and to identify some of the issues which needs to be addressed.**

Keywords: Instruction Set Extensions, AES, ECC

1. INTRODUCTION

Cryptography deals with secure data communication achieved widely by using symmetric key cryptography and public key cryptography algorithms. In symmetric key algorithms like DES (Data Encryption standard) and AES (Advanced Encryption Standard), secret key is shared between sender and receiver. Two separate keys namely public key and private key are used for encryption/ decryption in case of public key algorithms like RSA and Diffie-Hellman. Most of the secure data communication systems employ public key algorithms in the beginning of session to authenticate communicating parties and to exchange private key in a secure manner. Subsequently, symmetric key algorithms are used for secure communication.

Figure 1 shows encryption/ decryption in cryptographic algorithms.

The customization of an instruction-set presents many advantages among others: first, the application code can be more densely encoded, resulting in a code size reduction; second, the total number of instructions that have to be executed may be reduced, which results in a lower power consumption and third, the execution of the application can be more efficient in terms of increased performance using the customized instruction [1].

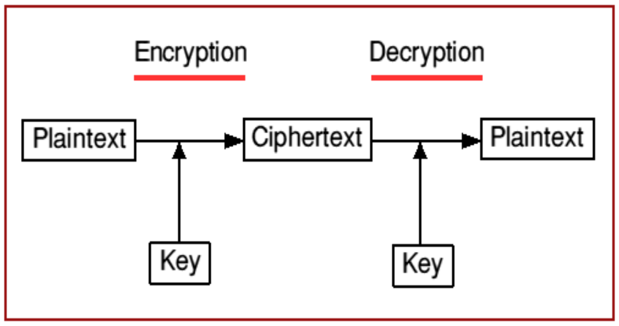

Fig. 1: Encryption/ Decryption

Cryptography can be implemented directly in hardware, as software routines, or as a combination of both hardware and software. Though software implementation is cost-effective but it has low performance as compared to hardware implementation like IDEA engine. Most of the cryptographic algorithms are based on three operational parameters namely block size, key size, and number of rounds [2].

The concept of instruction set extensions (ISE) has been employed very successfully in the domain of multimedia and digital signal processing. Recent research has also shown the benefits of instruction set extensions for public-key cryptography. Performance of any cryptographic algorithm largely depends on the efficient implementation of its associated operations. One area where instruction set extensions can play a very crucial role is ASIP (Application Specific Instruction Set Processors).

Instruction set extension for symmetric key algorithms mostly include instructions for information confusion and diffusion which is performed through operations like permutation, substitution, etc. AES (Advanced Encryption Standard), one of the symmetric key algorithms is implemented on both 32-

bit microprocessors and 8-bit microcontrollers. Instruction set extensions for cryptographic applications is mainly designed for use in embedded processors. So, its implementation must consider constraints like limited memory, power consumption, etc.

2. ISE FOR AES (ADVANCED ENCRYPTION STANDARD)

AES is the most studied symmetric key cryptographic algorithm and a number of instructions have been proposed by various researchers for performance improvement in this algorithm. Some of the instructions are explained in the following section.

2.1 AES-NI (Intel Advanced Encryption Standard New Instructions)

Intel AES New Instructions were introduced in 2010 for Intel Core processors. These instructions are designed for ensuring fast and secure data encryption and decryption. In this architecture, four instructions are used for encryption and decryption, and two instructions are used for key expansion. AES instructions also improve security of AES algorithm.

Table 1: AES-NI Instructions

Instruction Name	Description
AESENC	It Performs a round of an AES encryption
AESENCLAST	It Performs last round of an AES encryption
AESDEC	It performs a round of an AES decryption
AESDECLAST	It performs last round of an AES decryption
AESKEYGENASSIST	It assists in AES key generation and generates round keys used in expansion
AESIMC	It assists in AES key expansion by converting the encryption round key to be used for decryption

As shown in table 1, four instructions namely AESENC (AES Encrypt Round), AESENCLAST (AES Encrypt Last Round), AESDEC (AES Decrypt Round), and AESDECLAST (AES Decrypt Last Round) are provided for data encryption and decryption. They perform a sequence of transformations. AESKEYGENASSIST (AES Key Generation Assist) instruction generates round key used for encryption and AESIMC (AES Inverse Mix Columns) instruction converts encryption round key to be used for decryption by using inverse cipher.

Intel's AES instructions are flexible enough to support any number of rounds, any key length, and also various block sizes. These instructions improve performance while encrypting and decrypting large amount of data as compared to software only, lookup tables based AES implementations. Performance improvement using Intel AES-NI in different modes of operation, with 128, 192, and 256 bit key lengths is shown in table 2 [3].

Table 2: The Performance of AES Encryption/ Decryption of a 1 K Bytes Buffer, in Various Modes of Operation

	AES 128	AES 192	AES 256
	Performance in CPU Cycles per Byte for a 1 KB Buffer		
ECB Encryption	1.28	1.53	1.76
ECB Decryption	1.26	1.51	1.76
CBC Encryption	4.15	4.91	5.65
CBC Decryption	1.30	1.53	1.78
CTR Encryption/ Decryption	1.38	1.61	1.88

Source: Intel Corporation

2.2 CLMUL (Carry-less multiplication) Instruction Set

Carry-less multiplication is an essential operation in many cryptographic algorithms. Intel and AMD proposed Carry-less multiplication (CLMUL), an extension to x86 instruction set for their microprocessors in 2008. This instruction set appeared in westmere processor in 2010. It improves speed of applications performing block cipher encryption. The PCLMULQDQ instruction speeds up the carry-less multiplication of two 64-bit operands which is a relatively time consuming operation. It is especially important for implementing a recently defined mode of operation in block ciphers used with AES called GCM (Galois Counter Mode). Software implementations of GCM were not very efficient for existing IA instruction sets [4].

3. ISE FOR ECC (ELLIPTIC CURVE CRYPTOGRAPHY)

ECC is a public-key cryptography approach based on algebraic structure of elliptic curves over finite fields. ECC has the advantages of shorter keys which results in fast implementation thereby consuming less energy and bandwidth. One of the challenge in implementing ECC algorithm is that handling finite field in software is a complex task as field sizes may be too long. [5].

ISE for ECC provides good security with smaller key sizes which is very beneficial for certain applications. Following table shows that ECC algorithm has shorter keys as compared to RSA algorithm which makes it a better choice for cryptographic implementations [6].

Groszschaedl et al [7] proposed an approach for efficient arithmetic processing in Elliptical Curve Cryptosystems by providing new algorithms for multiple-precision arithmetic in GF (2^m). As compared to conventional software implementations, this approach is considerably faster.

Table 3: Key Strength Comparison of RSA and ECC

RSA Algorithm	ECC Algorithm
1024	160
2048	282
4096	409

4. HARDWARE IMPLEMENTATIONS

To implement AES-NI and CLMUL instructions, following hardware architectures are proposed:

4.1 Intel Westmere

Westmere is the family of processors implementing AES-NI and CLMUL instructions. It offers full hardware support for AES algorithm. This processor performs hardware-accelerated encryption which results in faster execution and protection against software targeted attacks. AES-NI implemented in westmere enables broad use of AES and it also improves security. Westmere is the successor of Intel's previous architecture named Nahalem. It is expected to deliver at least a threefold performance increase while performing cryptographic operations.

4.2 ARMv8 Architecture

ARMv8 architecture has instruction level support for cryptography. For implementing AES, it has 2 encode and 2 decode instructions. It works on advanced SIMD 128-bit registers. This architecture also supports SHA (Secure Hash Algorithm) using two 128 bit wide register. ARMv8 crypto instructions consist of instructions like PMULL (Polynomial multiply long vector), AESE (AES single round encryption), AESD (AES single round decryption, etc. for accelerating the task of encryption and decryption.

4.3 Intel Haswell architecture

Haswell microarchitecture introduced with 4th generation Intel Core processor family provided implementation for many new instructions proposed to improve performance in cryptographic applications. This architecture also provides better performance while implementing AES-NI instructions compared to previous Intel microarchitectures namely Sandy Bridge and Ivy Bridge. This architecture provides cryptographic algorithm implementations for AES, RSA, and SHA algorithms [8]. Performance gain from Haswell architecture for different modes of operation in AES algorithm as compared to previous Sandy Bridge architecture is shown in the table below:

Table 4: AES Performance (Cycles/Byte) 1

Algorithm	**Intel® microarchitecture code name Sandy Bridge**	**Haswell**	**Haswell Gain**
AES-128-CBC Encrypt	5.21	4.52	1.15x
AES-128-CBC Decrypt	0.76	0.64	1.19x
AES-256-CBC Encrypt	7.21	6.27	1.15x
AES-256-CBC Decrypt	1.02	0.89	1.15x
AES GCM Encrypt	2.76	1.26	2.19x

Source: Intel Corporation

5. ISSUES IN INSTRUCTION SET EXTENSIONS

Identifying new instructions for cryptographic applications is a complex task which is based on different parameters such as power consumption, code size, area, cycle count, etc. Also, all the instructions are not suitable to be implemented in hardware due to limited hardware resources. Another issue in generating custom instructions is the required human effort in identifying and implementing instruction set extensions. Limited number of input and output operands of the custom instructions is another limitation in instruction set extension [9].

Many of the existing microprocessors implementing cryptographic algorithms do not exploit parallelism to its full extent. Efficient implementation of operations and efficient use of clock cycles are the issues which need to be addressed while implementing cryptographic algorithms. Very few researchers attempted to provide a common architecture for supporting different public-key cryptosystems.

6. CONCLUSION AND FUTURE WORK

We presented a survey of newly proposed instructions implemented on processor microarchitectures supporting cryptographic applications. We have also reported various issues concerning instruction set extension for cryptographic domain. We have observed that there is a need for common architectural implementations which can effectively support different cryptosystems. Though there is a significant performance improvement in processing cryptographic algorithms through instruction set extensions, still there is a scope to further accelerate computing intensive operations in these algorithms by introducing new instructions. There is a need for methodologies that can automate the task of selecting complex instruction set extensions for application specific processors. ASIP's for cryptographic implementation can benefit a lot from instruction set extensions.

REFERENCES

[1] Arnold, M. and Corporaal , H., "Designing domain-specific processors", Proceedings of the ninth international symposium on Hardware/software codesign, 2001, pp 61–66.

[2] Lisa Wu, Chris Weaver, and Todd Austin, "CryptoManiac: A Fast Flexible Architecture for Secure Communication", Proceedings 28th IEEE Annual International Symposium on Computer Architecture, 2001, pp. 110-119.

[3] Shay Gueron, "Intel® Advanced Encryption Standard (AES) New Instructions Set", Intel Architecture Group, Israel Development Center, Intel Corporation.

[4] Shay Gueron, Michael E. Kounavis, "Intel® Carry-Less Multiplication Instruction and its Usage for Computing the GCM Mode ", Intel Corporation.

[5] Sandro Bartolini, Roberto Giorgi, and Enrico Martinelli, "Instruction Set Extensions for Cryptographic Applications ", Cryptographic Engineering, 2009, pp 191-233.

[6] H. Parveen Begam and Maluk Mohamed M. A., "Performance Analysis of Elliptic Curve Cryptography Using Onion Routing to Enhance the Privacy and Anonymity in Grid Computing", International Journal of Future Computer and Communication, Vol. 1, No. 2, August 2012.

[7] Groszschaedl, J., Kamendje, G.-A., "Instruction set extension for fast elliptic curve cryptography over binary finite fields GF (2m)", Proceedings IEEE International Conference on Application-Specific Systems, Architectures, and Processors, 2003.

[8] Sean Gulley, Vinodh Gopal, "Haswell Cryptographic Performance ", Intel Corporation, IA Architects, Intel Corporation.

[9] Carlo Galuzzi, Koen Bertels, "The Instruction-Set Extension Problem: A Survey", ACM Transactions on Reconfigurable Technology and Systems, 2010.