Benchmarking Message Authentication Code Functions for Mobile Computing

A. M. Rashwan[#1], A-E M. Taha[*2], and H.S. Hassanein[#3]

[#]Telecommunications Research Lab
School of Computing
Queen's University
Kingston, ON, Canada K7L 3N6
{[1]arashwan, [3]hossam}@cs.queensu.ca

[*]Electrical Engineering Department
Alfaisal University
P.O. Box 5092
Riyadh 11533 KSA
[2]ataha@alfaisal.edu

Abstract— **With the increased popularity of both Internet and mobile computing, several security mechanisms, each using various cryptography functions, have been proposed to ensure that future generation Internets will guarantee both authenticity and data integrity. These functions are usually computationally intensive resulting in large communication delays and energy consumption for the power-limited mobile systems. The functions are also implemented in variety of ways with different resource demands, and may run differently depending on platform. Since communications within the next generation Internet are to be secured, it is important for a mobile system to be suited to the function that provide sufficient communication security while maintaining both power-efficiency and delay requirements. This paper benchmarks mobile systems with cryptographic functions used in message authentication. This paper also introduces a metric, namely apparent processing, that makes benchmarking meaningful for mobile systems with multiple processing cores or utilizing hardware-based cryptography. In addition, this paper discusses some of evaluated functions' computational characteristics observed through benchmarking on selected mobile computing architectures.**

Keywords-component; message authentication code; message hashing; next generation Internet security; mobile security;

I. INTRODUCTION

Many standards and proposals have been made to ensure the authenticity and data integrity of Internet communications at different levels. Such proposals include IPSec, SSL/TLS, Kerberos, TCP-AO, SCTP-AUTH, to name a few [1]; targeting various services from secured web browsing to protection of intercontinental routing information, and systems from small embedded devices to large mainframes and data centers. Many of those proposals achieve both authenticity and data integrity of transmitted messages with the use of cryptographic functions to generate verification tags, known as *Message Authentication Code* (MAC), to be attached to their corresponding messages.

While the use of cryptography in computing is essential to ensure the security of the information being transmitted through the Internet, it is also known to be computationally challenging. As the importance of security increases with the development of wireless communications and smart portable systems, researchers in security have continued developing, optimizing, and evaluating the performance of various cryptographic approaches. The major focus is finding the approach with strongest security measures and least possible resources demand. As a result, several cryptographic functions had found their ways as software-based and hardware-based solutions to provide message encryption and verification services for various systems and applications. These functions were subject to several research studies evaluating their security and performance, in terms of either the computational power or data throughput.

However, previous studies [2] [3] [4] [5] were evaluating the absolute computational or resource demands of a group of selected functions running on the architecture of study. Only slight considerations were made for how functions behave under dynamically-controlled resource limitations. Cryptographic functions are further attached to other processes, such as communication sessions, and not operated in isolation. Thus, evaluating them for communication purposes should be conducted in a scenario that reflects existence of multiple sessions competing on the existing resources, instead of simply benchmarking for the absolute performance or throughput on the system of study. Especially if the evaluated function is not processed by the same processor where communications are handled – a common setup for many mobile systems [6].

This paper investigates a more comprehensive way to evaluate cryptographic functions that offer MAC services for mobile communications. In this paper, the computational characteristics of some known MAC functions are observed through benchmarking them for communication purposes; under different mobile computing architectures. In doing so, we suggest the use of a new metric, called apparent processing, to facilitate a meaningful comparison between MAC functions. The metric borrows from the notion of apparent parallelism utilized in the context of parallel computing.

The remainder of this paper is organized as follows. Section II refers to some of the related works and describes motivations behind inducting the benchmarking of MAC functions. Section III defines the metric, assumptions, run environments, and used workload for the benchmarking. Evaluation of the benchmarking results is presented in Section IV. Section V lists some additional observed considerations and challenges when benchmarking MAC functions. Finally, conclusion and future directions are mentioned in Section VI.

II. RELATED WORK AND MOTIVATION

MAC functions are powered by two types of cryptographic functions: hash or block-cipher [7]. Hash functions, such as MD5, SHA1, and SHA2, are one-way compression algorithms that map variable-length larger messages into shorter fixed-sized strings that vary for different messages. Block-cipher functions, such as AES, TWOFISH, SERPENT, and RC6, are encryption/decryption algorithms designed to work on fixed-sized portions of given messages, called blocks. MACs are generated either by directly hashing combined messages with provided secrets using hash functions, or by hashing message blocks encrypted with provided secrets when using block-cipher-functions.

There are many performance studies of both hash and block-cipher functions in the literature; evaluating different implementations directly and indirectly for various applications [8] [9], architectures [2] [5] [10] [11], and network structures [3] [12]. Those studies were evaluating either the processing power of evaluated functions on certain processing units, or the data throughput that the evaluated functions process.

Today's mobile systems are powered by various single-core and multiple-core processors of different architectures. In addition, several mobile systems are equipped with additional hardware-assisted components to offload computing intensive operations, such as graphics and cryptography, from their main processors, while communications are still maintained by the main processors. Different considerations must be taken into account when benchmarking MAC functions for network communications. These considerations include:

- **Cores Involved.** The future generation Internet protocols, and the future production operating systems in response, will have message authentication as a standard feature. Execution of MAC functions might be locked on certain processor cores so those functions do not render the system unusable in case of high load. Previously conducted performance studies usually focused on the processor power consumed by an evaluated function, without taking into consideration how many cores are utilized by that function.
- **Extra-Processor Computation.** There are implementations of MAC functions that execute outside the main processor (e.g. an external cryptographic processor), yet the main processor still responsible for handling the communication session utilizing the MAC function. Evaluating the processing power on either the main processor or the external processor alone might not be sufficient to indicate the effective performance of that communication session.
- **Implementation Heterogeneity.** A MAC function can have different implementations under same computing architectures, for example, single- threaded and multi-threaded implementations. The determination of superior implementation will be subject preliminary to how the processor will handle them, and how the operating system will schedule their execution. Other factors such as I/O and bus delays may also contribute to the determination.
- **Workload Concurrency.** Previous studies did not consider the use of concurrent workloads (for example, to simulate multiple connections) for their evaluations. With concurrent workloads, effects of scheduling, memory and I/O demands can be reflected on the processing time, giving more realistic performance determination (such as computing the system' capacity in term of number of concurrent connections).

It becomes obvious that there are substantial considerations to be made beyond power or throughput. The objective of this work is to realize an environment to characterize the effect of these considerations.

III. Benchmarking setup

In order to successfully evaluate the MAC functions for communication purposes within mobile systems, it is important to have a benchmarking setup that effectively describes the true nature of both mobile environments and communications. To achieve this, the benchmarking setup should incorporate the use of mobile production systems, and production operating systems configured for everyday usage. For communications, the setup should be simulate concurrent processing of message authentication as it happens with real communication sessions, and to operate on realistic message lengths that are supported by major mobile communication protocols.

In this paper, the benchmarking setup was designed to fulfill the mentioned objectives. Real architectures from three major known brands in mobile computing were evaluated. A customizable distribution of Linux operating system, which its kernel is powering many of today's mobile systems, is used. The workload was implemented to apply message authentication processing in simulated communication mode, while controlling some operational factors such as cores involved, number of concurrent sessions, and session duration. The workflow was also designed to ensure that the both workloads and their parameters were suitable to apply on studied systems for accurate evaluations. The benchmark metrics were selected to observe computational characteristics of studied systems in both simple and comprehensive ways.

The following elaborates on the details of the benchmarking setup.

A. Benchmark Environment

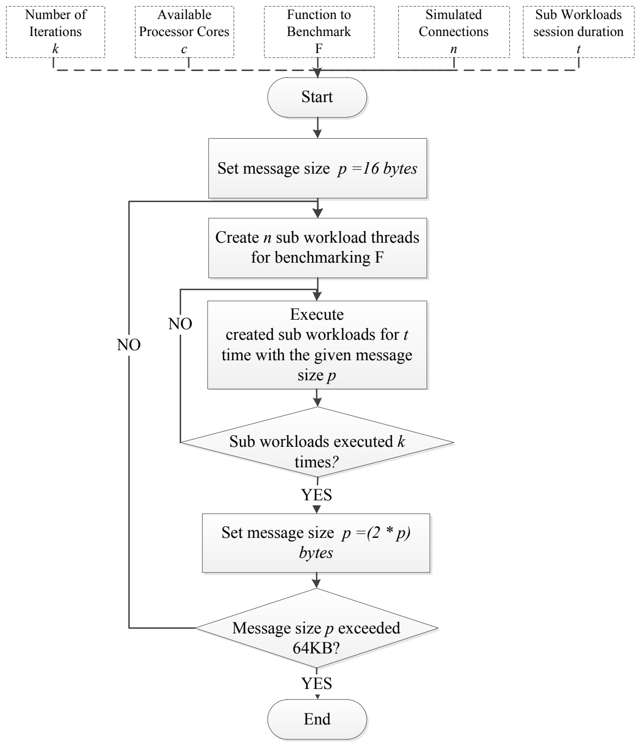
The environment is designed in a manner that ensures that the operating system has minimum influence on obtained results, all benchmarking experiments were conducted under the same OS (Ubuntu Linux 10.04LTS). The chosen operating system, which is running in Gnome desktop mode, uses "Completely Fair" scheduler for scheduling its processes.

Benchmarked architectures were TI DM3730 ARM Cortex A8, Intel Core I3 M350, AMD Opteron 2354, Mobile Intel Pentium 4 3.0GHz. While Opteron is not build for mobile systems, it was included in the benchmark since it was the available AMD architecture at the time of the study, as it shares similar features with its mobile counterpart (Phenom) such as *Cool'n'Quiet*^{TM} speed-stepping (which is a mobile-based feature that reduces processor clock frequency to save energy or reduce operating temperature).

B. Workload and Workflow

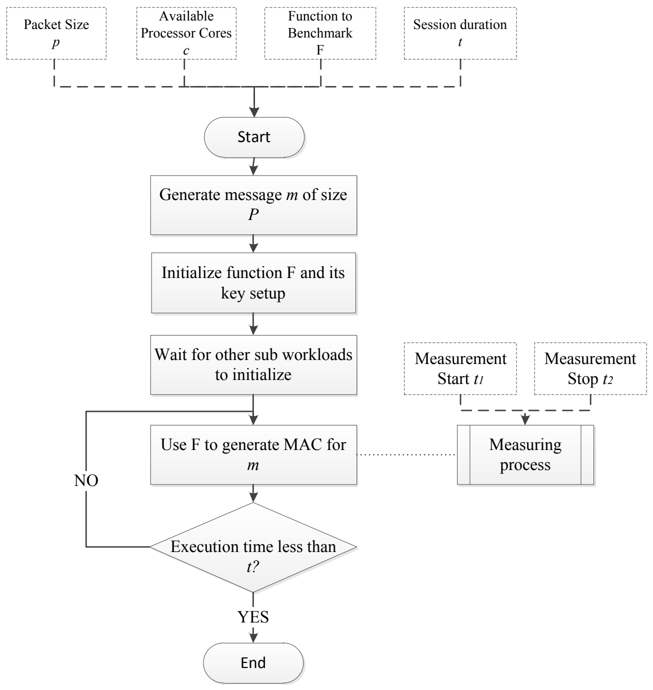
A multi-threaded benchmarking application is written to evaluate selected MAC functions. The application uses multithreading approach to create workload instances in order to minimize the effect of memory switching on the measurements accuracy. The application was also equipped with a method for binding the execution of the workload instances into certain predefined cores in order to study the effect of limiting the number of available processor cores on the performance of the workload.

The benchmarking workload procedure is illustrated in Figs. 1 and 2. Since typical communication sessions usually runs concurrently with different durations and message lengths, the workload was implemented to simulate such conditions;



Number of Iterations k
Available Processor Cores c
Function to Benchmark F
Simulated Connections n
Sub Workloads session duration t
Start
Set message size p =16 bytes
Create n sub workload threads for benchmarking F
Execute created sub workloads for t time with the given message size p
NO
NO
Sub workloads executed k times?
YES
Set message size p =(2 * p) bytes
Message size p exceeded 64KB?
YES
End

Fig. 1. Benchmark workload procedure



Packet Size p
Available Processor Cores c
Function to Benchmark F
Session duration t
Start
Generate message m of size P
Initialize function F and its key setup
Wait for other sub workloads to initialize
Measurement Start t1
Measurement Stop t2
Use F to generate MAC for m
Measuring process
NO
Execution time less than t?
YES
End

Fig. 2. Benchmark sub workload procedure

with assumptions of fixed selection of evaluated function and fixed session duration per run to simplify the evaluation. In addition, workload executes the message authentication process same as with a typical production message authentication session. Under each studied architecture, the workload was applied with no other foreground applications running except for the Gnome desktop environment in order to reduce the effect on the measurement accuracy. The workload creates *n* sub workload threads to simulate existence of *n* connections transmitting messages ranging from 16 bytes to 64 kilobytes (the maximum length supported by the IP protocol). For each sub workload, there are *n*-1 background sub workloads representing *n*-1 background connections. In sub workloads, measurements are taken using both the processor clock and the system's wall clock. The period of taking measurements is defined by interval from t_1 to t_2, where t_1 and t_2 are predefined time values between 0 and the execution time *t, with* $t_2 > t_1$.

Fig. 3 describes the benchmarking workflow procedure. This procedure is essential to ensure workloads being applied to suit benchmarked systems, and measurements obtained are accurate and informative. First, a benchmark scenario is defined to include the number of connections, the function to evaluate, and the processor cores to utilize. Through the process, obtained measurements are analyzed and validated. For example, the accuracy can be affected by resources overutilization, insufficient session duration for reaching the steady state, or inappropriate measurement period. In this case, their corresponding parameters, which are the number of connections *n*, the session duration *t*, and the measurement period $[t_1,t_2]$ will be adjusted accordingly for the next workload.

C. *Evaluated MAC functions*

1) Hashed-based MAC (HMAC) [13]: a hash based algorithm, which is being used by various popular Internet protocols (e.g IPSec and TLS [1] [14]) to offer message authentication service during communication sessions Evaluated hash functions under this group are MD5, SHA1, SHA2-256 (simply SHA256) and SHA2-512 (simply SHA512).

2) One-key MAC One (OMAC1): also known as CMAC; a block-cihper based algorithm that was introduced to resolve security flaw of it predecessor, CBC-MAC, when generating MACs for variable-length messages [15]. In 2005, NIST recommended the using of OMAC1 for operating a block-cipher based authentication [16]. Evaluated block-cipher functions under this group are RIJNDAEL, TWOFISH, SERPENT, and RC6, which. were the finalist candidates in *Advanced Encryption Standard* (AES) selection process with Rijndael becoming the official AES [2].

3) VMAC: a block-cipher based algorithm designed to offer high-performance message authentication service [17]. Optimized for 64-bit computing architectures, VMAC utilizes block cipher functions via a "universal hashing" algorithm and secret key to generate MACs for a given message. Evaluated

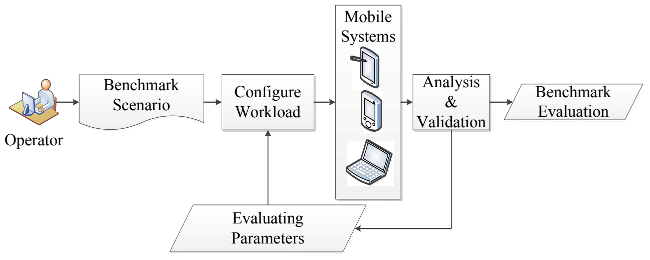


Fig. 3. Benchmark workflow procedure

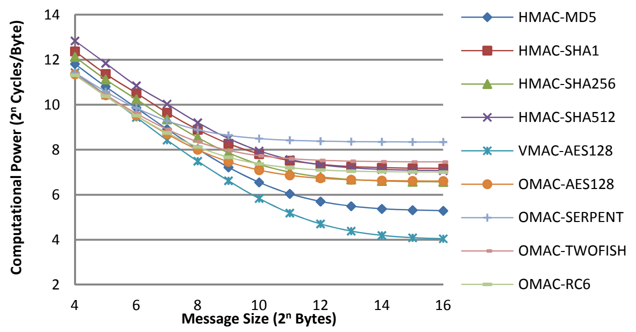


Fig. 4. Absolute Processing Power vs Message Size for different MAC functions under AMD Opteron 2354 @ 1.1GHz with all cores utilized (hardware-assisted compilation of CryptoPP)

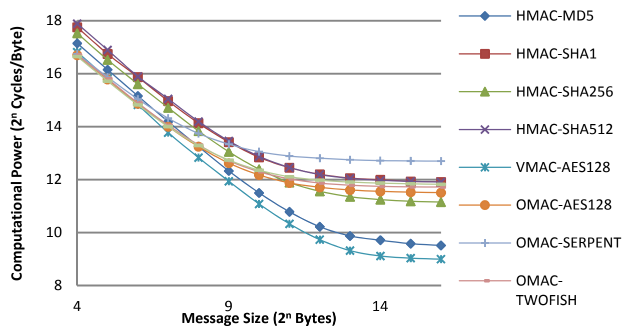


Fig. 5. Absolute Processing Power vs Message Size for different MAC functions under Mobile Intel P4 @ 3.0GHz with hyper-threading disabled (hardware-assisted compilation of CryptoPP)

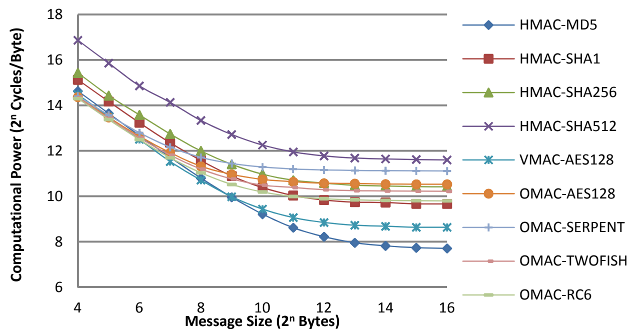


Fig. 6. Absolute Processing Power vs Message Size for different MAC functions under Mobile Intel Core I3 M350 @ 2.26GHz with all cores utilized (software-only compilation of CryptoPP)

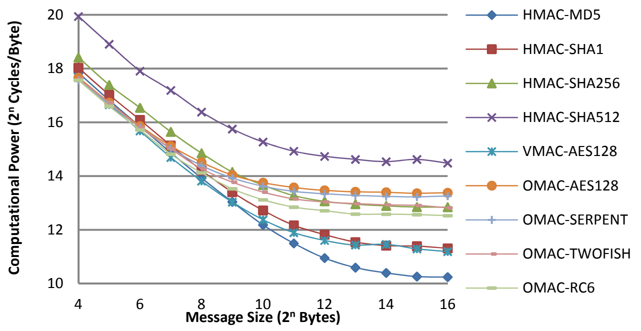


Fig. 7. Absolute Processing Power vs Message Size for different MAC functions under TI DM3730 ARM Cortex A8 @ 1GHz (software-only compilation of CryptoPP)

block-cipher function under this group is AES, which is the only implemetation avaliable for VMAC at the time of study.

D. *Cryptograhic Library Used in Benchmarking*

A well-known cryptographic library called CryptoPP 5.6.1 is used for this study. This selection is motivated by its popularity among academia for studying cryptographic performance and cryptanalysis [3]. It is also open-source and has cross-platform compatibility, making it suitable to run under various operating systems and computer architectures. The library further has both hardware-assisted and software-only implementations for some functions such as AES (using x86 AES-NI extension) and SHA-256/512 (Using x86 SSE-2 extension); making it a good option to test the effect of different implementations under same hardware. More significantly, the library has its own benchmarking tool that can be used as a validation tool for our benchmarking results.

E. *Metrics*

Cryptographic functions are known to be computationally demanding, but many of them do not pose high memory and I/O demands; making the processing power the only reasonable evaluation metric. In this paper, two types processing power metrics are considered. The first type presents the choice of most previous studies for evaluating cryptographic functions. While the second type is being introduced in this paper to show the effective performance on mobile systems through evaluating cryptographic functions at external or several processing units. For both types, the less value they have the better performance they indicate.

1) *Absolute Processing Power (presented in cycles/byte):* the number of processor cycles that a MAC function spent on main processor coding one byte of date. This metric is obtained directly from the main processor clock.

2) *Apparent Processing Power (presented in cycles/byte):* the wall clock time, calculated as the number of processor cycles, that a MAC function spent coding one byte of data, as if the function is virtually being executed squentially by the main processor. This metric is introduced for evaluating both multiple-core and hardward-assited systems (adopted from *apparent parallism* defined in [18] for symetric mutilple processor systems).

F. *Measuring Assumptions*

The following assumptions were made in evaluating the performance of the different MAC functions.

- Only the processing power is measured. Memory usage and I/O demands are not included in this study. However, the apparent processing power should reflect the memory and I/O demands since both memory transfers and I/O waits take time.
- Only the time for generating MACs is measured. Key setup time is considered in this study.
- Measuring overhead is taken into consideration when setting up the workload to minimize its effect on measurement accuracy.

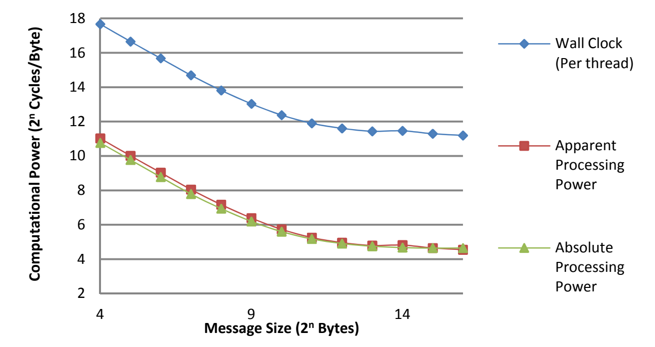


Fig. 8. Wall Clock Cycles,Absolute Processing Power, and Apparent Processing Power of VMAC-AES128 per thread when applying 100-thread workload on TI DM3730 ARM Cortex A8 @ 1GHz

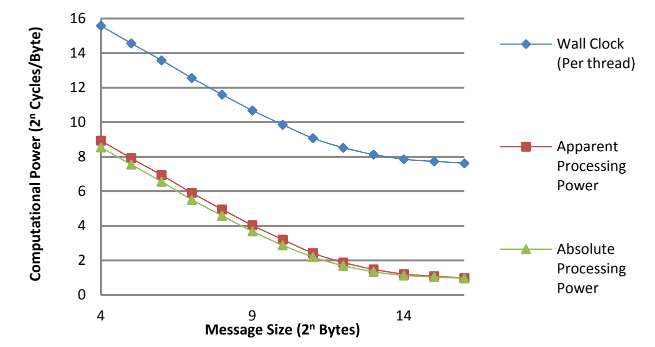


Fig. 9. Wall Clock Cycles,Absolute Processing Power, and Apparent Processing Power of VMAC-AES128 per thread when applying 100-thread workload on one core of Intel Core I3 M350 @ 2.26GHz

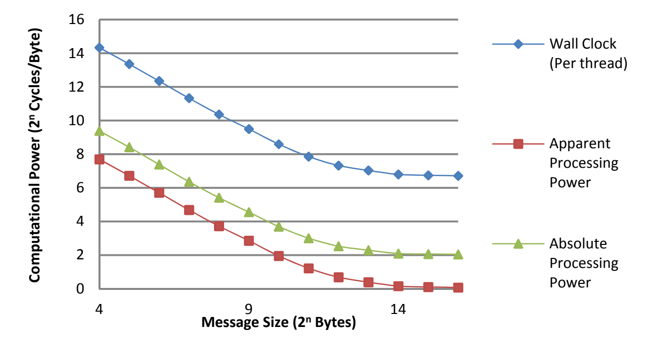


Fig. 10. Wall Clock Cycles,Absolute Processing Power, and Apparent Processing Power of VMAC-AES128 per thread when applying 100-thread workload on both cores of Intel Core I3 M350 @ 2.26GHz

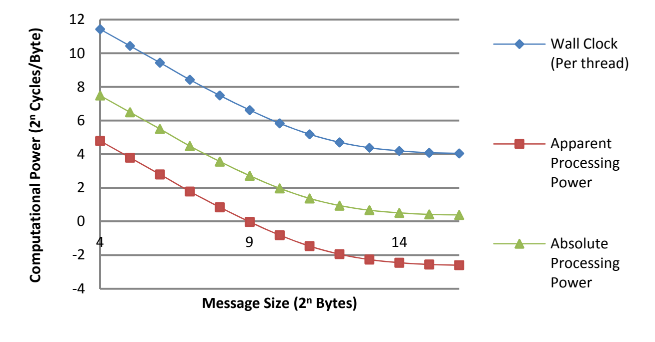


Fig. 11. Wall Clock Cycles,Absolute Processing Power, and Apparent Processing Power of VMAC-AES128 per thread when applying 100-thread workload on all eight cores of AMD Opteron 2354 @ 1.1GHz

- Effect of mobility features, such as speed stepping and thermal protection, on processing delays is not considered in this study.

IV. Results

A. *Behaviour of Different MAC generating functions under Different Architectures*

Figs. 4, 5, 6 and 7 show the absolute processing power for the evaluated functions under different architectures, and with 100 simultaneous threads per workload.

1) General computational trend for achitectures using the same function implementation: the computational trend, using the same compilation for CryptoPP library, is almost the same across evaluated architectures. The slight differences in performance between different architectures running the same compilations are due to how different processors handle and optimize execution of programs.

2) Computational trend verses message size: it is obvious that the computational trend changes with message sizes under different architectures. Thus, systems and services can optimize the selection of the best performing function based on the size of the transmitted message; provided that other factors such as security stength and key setup time are also taken into consideration.

3) Effect of a function's implementation on computational trend: In the hardware-assisted compilation of CryptoPP, as shown in Figs. 4 and 5, functions such as AES and SHA take advantage of speed boosting. While in Figs. 6 and 7 both AES and SHA lost their advantage, and thus the computational trend has changed with HMAC-MD5 being the fastest processing 64 kilobyte messages, and HMAC-SHA512 being the slowest (as opposed to VMAC-AES128 for fastest and OMAC-SERPENT for slowest in the hardware-assisted compilation).

B. *Effect of the number of utilized processor cores*

1) Single-core verses multiple-cores architectures: Figs. 8, 9, and 10 show the absolute and the apparent processing power when running a 100 connections VMAC-AES128 workload. For single-core architectures, the apparent processing power should mirror the absolute one, or be slightly higher due to resources demand and measurement overheads. For multiple-core architectures, such as in Figs. 10 and 11, the apparent processing power reflects the effect of parallelism on the performance. As expected, the apparent performance is correlated to the number of cores utilized within the architecture, with additional cores translated to better apparent performance.

2) Utilizing single-core verses multiple-cores: While the apparent performance improved in Fig. 10 (compared to Fig. 9) due to the parallelism effect, the absoluste performance is dropped. One explanation is because the contention between the workload and the operating system, causing the latter to preempt execution of the workload more frequenty. In addition, with the increase of the number of cores, the overhead caused by thread migration between cores might increase as well.

V. Observed Challenges

We note some challenges that were observed through conducting the benchmarking experimenting and workflow procedures. These challenges affect the accuracy of measuring both absolute and apparent processing powers, or in some cases, render them useless as evaluation metrics.

A. *Multi-threaded processor architectures*

Several computing architectures support execution of two or more threads per core (known as multi-threading cores). Operating systems usually treat each thread as a separate logical core, with no possibility of differentiating it from a physical core. Thus evaluating multi-core systems through limiting available cores can represent a challenge, especially if their main processors have three or more physical cores.

B. *MAC function offloading outside the main processor*

Some systems might offload MAC functions outside their main processor. Examples of places where MAC functions are offloaded into include *General Purpose Graphic Processing Unit* (GPGPU) and cryptographic processing units. While the performance can be evaluated from the offload unit; it is important not to ignore the main processor in the evaluation since it is still handling communications secured by the offloaded MAC function. In such cases, it will be more logical to benchmark for the apparent processing performance, as if the main processor is executing the functions, although, it will not point to the absolute performance of the evaluated MAC function at the offload unit.

C. *Systems with no synchronized clock accross all of its processors or cores*

In a multiple processors/cores system, a thread can jump from one processor/core to another. If such a system does not maintain a synchronized clock across all the processors/cores, it is not possible to get the absolute performance for the evaluated MAC function. It will still possible, however, to benchmark for the apparent performance. An estimate for absolute performance can then be made knowing the system's parallelism factor.

D. *Operating System Scheduling*

It is important to be aware of the scheduling algorithm used by the evaluated operating system. Some operating systems might schedule sub workloads unevenly, which will be mainly reflected in the measured apparent performance. For example, least scheduled threads will have higher apparent processing power than the most scheduled threads of same workload.

VI. Conclusions

Message Authentication Codes (MAC) functions with similar implementations perform with nearly similar computational trends regardless of the computing architecture or its number of cores, the trend depends on the message size. This finding is elemental in selecting the MAC function based on message size, delay, energy, and security requirements. It should also be noted that utilizing all processing cores within a system increases the absolute processing power of MAC functions due to resource contention with the operating system and to overhead of process switching between cores. Systems may therefore employ methods to allocate the number of cores in a manner that optimizes both absolute and apparent processing performance.

The use of the apparent processing power metric for benchmarking seems highly useful when evaluating MAC functions in systems with multiple processors/cores, systems with multi-threaded cores, and systems with offload cryptographic processing units. While the apparent processing performance does not always reflect the absolute performance of the evaluated MAC function, it can be considered as an effective performance metric for communication purposes. However, more investigations are required for finding a feasible relation between the apparent performance and some other factors, such as the used task scheduling and the physical resource requirements of the evaluated MAC function.

Acknowledgment

This research is funded by a grant from Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] Internet Official Protocol Standards. RFC 5000, IETF, May 2008, http://www.ietf.org/rfc/rfc5000.txt.

[2] B. Schneier and D. Whiting, "A performance comparison of the five AES finalists," , 2001, p. the 3rd AES Conference (AES3).

[3] Y. W. Law, J. Doumen, and P. Hartel, "Survey and Benchmark of Block Ciphers for Wireless Sensor Networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 2, pp. 65-93, February 2006.

[4] D. S. Abd Elminaam, H. M. Abdual Kader, and M. M. Hadhoud, "Evaluating The Performance of Symmetric Encryption Algorithms," *International Journal of Network Security*, vol. 10, no. 3, pp. 216–222, 2010.

[5] O. Hyncica, P. Kucera, P. Honzik, and P. Fiedler, "Performance evaluation of symmetric cryptography in embedded systems," in *IEEE 6th International Conference on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS)*, 2011, pp. 277-282.

[6] R. Anderson, M. Bond, J. Clulow, and S. Skorobogatov, "Cryptographic Processors - A Survey," *Proceedings of the IEEE*, vol. 94, no. 2, pp. 357-369, Feb. 2006.

[7] C. Paar, J. Pelzl, and B. Preneel, *Understanding Cryptography: A Textbook for Students and Practitioners*.: Springer, 2010.

[8] M. Sokol, S. Gajewski, M. Gajewska, and L. Staszkiewicz, "Security and Performance Analysis of IPsec-based VPNs in RSMAD," in *The First International Conference on Advanced Communications and Computation (INFOCOMP'11)*, 2011, pp. 70-74.

[9] C. Shen, E. M. Nahum, H. Schulzrinne, and C. Wright, "The impact of TLS on SIP server performance," in *IPTComm'10*, 2010, pp. 59-70.

[10] D. J. Bernstein and P. Schwabe., "New AES software speed records," *INDOCRYPT*, vol. 5365 of LNCS, pp. 322-336, 2008.

[11] D.A. Osvik, J.W. Bos, D. Stefan, and D. Canright, "Fast software AES encryption," *Fast Software Encryption (FSE)*, vol. 6147 of LNCS, pp. 75–93, 2010.

[12] J. Kaps and B. Sunar, "Energy Comparison of AES and SHA-1 for Ubiquitous Computing," in *Embedded and Ubiquitous Computing*, 2006, pp. 372-381.

[13] H. Krawczyk, M. Bellare, and R. Canetti, HMAC: Keyed-Hashing for Message Authentication. RFC 2104, IETF, Feb. 1997, http://www.ietf.org/rfc/rfc2104.txt.

[14] T. Dierks and E. Rescorla, The Transport Layer Security (TLS) Protocol Version 1.2, 2008, http://www.ietf.org/rfc/rfc5246.txt.

[15] Kaoru Kurosawa Tetsu Iwata, "OMAC: One-Key CBC MAC," *Fast Software Encryption (FSE)*, pp. 129-153, 2003.

[16] M. Dworkin, Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, May 2005, NIST Special Publication 800-38B.

[17] T. Krovetz, "Message authentication on 64-bit architectures," in *13th international conference on Selected areas in cryptography* , 2006, pp. 327-341.

[18] Eli Biham, *Fast software encryption: 4th International Workshop, FSE '97*.: Springer, 1997.