Accelerated Processing of Secure Email by Exploiting Built-in Security Features on the Intel® EP80579 Integrated Processor with Intel® QuickAssist Technology

Vallisha Keshavamurthy and Shambhu Upadhyaya
Computer Science and Engineering
University at Buffalo
Buffalo, NY 14260
{vallisha,shambhu}@buffalo.edu

Vinodh Gopal
Intel Corporation
75 Reed Road
Hudson, MA 01749
vinodh.gopal@intel.com

Abstract – *Domain Keys Identified Mail (DKIM) is one of the widely used mechanisms by which email messages can be cryptographically signed, permitting a signing domain to claim responsibility for the release of an email into the mail stream. As the volume of emails exchanged becomes large, the software implementations of DKIM using OpenSSL library will become a limiting factor of performance due to the heavy computations involved. In this largely empirical work, we identify the computation intensive modules of DKIM and solve the performance issues by implementing their functions on COTS hardware. Our approach makes use of the Intel Embedded processor Tolapai (Intel EP80579) that has several built-in cryptographic functionalities, viz. security accelerators for bulk encryption, authentication, hashing and public/private key generation and digital signing. Experimental results show that an overall 50% acceleration can be achieved by transparently migrating the DKIM functionalities to hardware.*

Keywords – Domain Keys Identified Mail, Hardware security, Hashing, Intel EP80579, Secure email

I. INTRODUCTION

A common synonym for spam is unsolicited bulk email (UBE). The definition of spam usually includes the aspect that an email is unsolicited and sent in bulk. Spam is one of the media for fraudsters to scam users to enter personal information on fake websites using email forged to look like it is from a bank or other legitimate organization. This is known as phishing [1].

According to IronPort's 2008 Security Trend Report, as much as 90% of inbound mail is spam today. Moreover, spam is no longer simply an irritant but becomes increasingly dangerous. About 83% of spam contains a clickable link. Thus, phishing sites and Trojan infections of office and home systems alike are just one click away. State-of-the-art spam filtering techniques are based on content analysis (e.g., SpamAssassin), host reputation (SpamCop, Spamhaus) or authentication services (SPF, DKIM) [2].

Our goal in this research work is to utilize a hardware accelerating SOC processor which will improve the performance of the domain keys identified mail (DKIM) algorithm processing and in-turn the DKIM based spam filters and phishing attack detectors. Such enhancements will facilitate the speedy processing of large volumes of emails exchanged at the gateways. We achieve our goals by moving to hardware the computation-intensive hashing functions and digital signature schemes using RSA and SHA which are part of the core DKIM algorithm. This paper is not about designing a new algorithm for DKIM based spam filters. On the other hand, it achieves acceleration for existing DKIM based spam filters and phishing attack detectors through an empirical study and implementation.

We performed this empirical study by considering two benchmarking methods with the goal of showing significant performance improvement of DKIM using hardware acceleration as compared to its software counterparts. The first one is the OpenSSL's built-in benchmark. This benchmark tests the performance of the crypto library used by DKIM. Second, we benchmark the response time of an end to end implementation of DKIM written in C. The results briefed out in sections IV and V show a significant performance improvement by implementing DKIM functions on hardware.

The organization of the paper is as follows. Section II describes the related work on DKIM and milters (stands for mail filters, which are used for filtering spam or viruses very efficiently in the mail-processing chain) and earlier approaches to move the milter functionalities to hardware. Section III details the Intel processor, the DKIM approach and the acceleration techniques. The experimentation that was carried out to illustrate our performance goals is described in Section IV. The results, conclusions and future work are briefed out in sections V and VI respectively.

II. RELATED WORK

Luo [14] gives details on a plethora of research on spam detection, filtering, elimination and some anti-spam appliances that have been introduced to the market. The DKIM algorithm and its scope and usage are explained in

[4], [6], [7] but there is no literature available on any hardware approaches for accelerating the performance of DKIM.

However, the growth of spam messages remains rampant. There exists a strong call to design high-performance email filtering systems. Most of the existing research focuses on the design of protocols, authentication methods, neural network based self-learning and statistical filtering. In contrast, we approach the spam filtering issues from a complementary perspective – improving the filtering performance through the computer architecture support.

Gupta et al. [1] introduce a technique which improves the performance of the Naïve Bayesian spam filters and phishing attack detectors by moving hashing functions used in Naïve Bayesian spam filters to Intel Tolapai (EP80579) hardware. There have been attempts to patch OpenSSL, the popular library used for DKIM implementations [11] and to use the OCF driver on Linux which enables it to accelerate cryptographic operations using the integrated cryptographic accelerator of Intel Tolapai. Our approach is to use the same multi-purpose cryptographic processor of Intel to achieve performance improvement of DKIM protocol and in-turn the DKIM based spam filters and phishing attack detectors. Being a multi-purpose processor, Intel Tolapai (EP80579) can run like a normal desktop processor and at the same time make use of its accelerating capabilities for security applications.

III. EXPERIMENTAL BASIS

We are using Intel EP80579 (Tolapai) processor to achieve acceleration of DKIM implementation using the OpenSSL library at its core. The details are described next.

A. Intel EP80579 Processor

The Intel® EP80579 Integrated Processor with Intel® QuickAssist Technology, Tolapai, is a complete System-on-a-Chip for security, communications, storage and embedded designs.

a) Architectural Details of Tolapai

The Intel® EP80579 Integrated Processor (Tolapai) is a System-On-a-Chip (SOC) integrating the Intel® Architecture core processor, the Integrated Memory Controller Hub (IMCH) and the Integrated I/O Controller Hub (IICH) all on the same die. In addition, it has integrated Intel® QuickAssist Technology, which provides acceleration of cryptographic and packet processing. Fig. 1 shows the architecture of Intel EP80579.

The Intel® QuickAssist Technology components housed in the Acceleration and I/O Complex (AIOC) are as follows:

- The Security Services Unit (SSU) provides acceleration of cryptographic processing for the most common symmetric cryptographic algorithms (ciphers such as AES, 3DES, DES, (A)RC4, and messages digest/hash functions such as MD5, SHA-1, SHA-2, HMAC, etc.), asymmetric cryptographic functions (modular exponentiation to support public key encryption such as RSA, Diffie-Hellman, DSA), and true random number generation.
- The Acceleration Services Unit (ASU) includes packet processing acceleration engines.

We utilize this acceleration capability of Intel EP80579 to improve the performance of DKIM implementation. The RSA and hashing functions as identified in Section III.C are moved to hardware as specified in Section III.C.b.

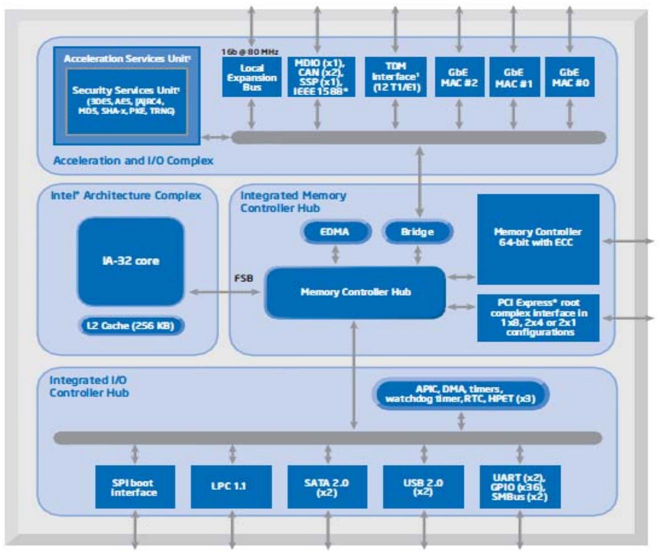

Acceleration Services Unit¹
Security Services Unit¹ (3DES, AES, (A)RC4, MD5, SHA-x, PKE, TRNG)
16b @ 80 MHz
Local Expansion Bus
MDIO (x1), CAN (x2), SSP (x1), IEEE 1588*
TDM Interface¹ (12 T1/E1)
GbE MAC #2
GbE MAC #1
GbE MAC #0
Acceleration and I/O Complex
Intel® Architecture Complex
IA-32 core
L2 Cache (256 KB)
FSB
Integrated Memory Controller Hub
EDMA
Bridge
Memory Controller Hub
Memory Controller 64-bit with ECC
PCI Express* root complex interface in 1x8, 2x4 or 2x1 configurations
Integrated I/O Controller Hub
APIC, DMA, timers, watchdog timer, RTC, HPET (x3)
SPI boot interface
LPC 1.1
SATA 2.0 (x2)
USB 2.0 (x2)
UART (x2), GPIO (x36), SMBus (x2)

Fig. 1: Block Diagram of Intel EP80579 [3]

b) Features of Tolapai

This SOC processor delivers a significant leap in architectural design, with a good combination of performance, power efficiency, footprint savings and cost-effectiveness compared to discrete, multi-chip solutions. Using multi-chip solutions for different security applications poses scalability and cost issues. Tolapai aims to provide a single chip solution for security applications. The integrated accelerators in this SOC processor support Intel QuickAssist Technology through software packages provided by Intel. These software packages provide the library structures to integrate security functionality into the application, completely adjunct to the Intel architecture complex, freeing up CPU cycles to support additional features and capabilities. This provides the efficiency of customized hardware with the flexibility to design diverse applications with one platform. The design also includes security accelerators for bulk encryption, hashing and public/private key generation [3].

Currently, the FWA-3240 [15], [16], which is a board-level product developed by Advantech Co., Ltd., incorporates Intel's Tolapai System-on-Chip which combines Intel's QuickAssist Technology and integrates an Intel Pentium M class core, memory controller and I/O controller. The high-performance CPU core supplies the horse power needed to perform deep packet inspection and other complex operations and is particularly optimized for entry to mid-range network security appliances.

B. DKIM Algorithm

Domain Keys Identified Mail (DKIM) is an anti-spam approach that involves digitally signed email [7]. Stephen et al. [7], Barry et al. [6] and Allman et al. [4] have provided the details about the DKIM algorithm and the scope and usage of it. The sender signing practices are explained by Allman et al. [5]. Fig. 2 shows the working of the DKIM algorithm. The flow is as follows:

- The sending domain publishes in its DNS record a public key (e.g., generated using OpenSSL).
- The sending mail server then digitally signs, using the private key, and sends the message.
- The receiving mail server verifies the digital signature by retrieving the public key of the sending domain from the DNS.
- The receiving mail server verifies the digital signature, and if successful, delivers the email to the end user.

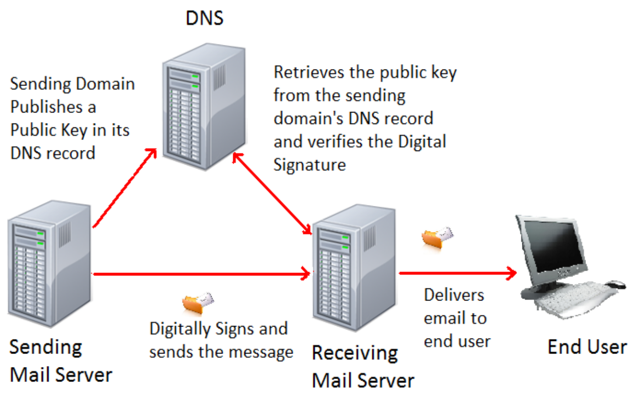

DNS
Sending Domain Publishes a Public Key in its DNS record
Retrieves the public key from the sending domain's DNS record and verifies the Digital Signature
Sending Mail Server
Digitally Signs and sends the message
Receiving Mail Server
Delivers email to end user
End User

Fig. 2: The workings of DKIM [8]

C. Achieving Acceleration

The DKIM feature is implemented through a milter (mail filter), which is an extension to the widely used open source mail transfer agent (MTA) Sendmail and Postfix. It allows administrators to add mail filters for filtering spam or viruses very efficiently in the mail-processing chain [17]. For example, Sendmail, Inc. has released a free, Open Source implementation of the DKIM signing and verifying software. A portable API is available to allow DKIM to be embedded into any application. Also provided is a milter plugin for Open Source sendmail, the world's leading MTA. The milter plugin allows system administrators to easily sign and verify messages using DKIM signatures [9]. As may be noted from above section the DKIM algorithm uses RSA public key cryptosystem along with SHA-1 and SHA-256 hashing at its core, which is provided using OpenSSL library utility by most of the popular DKIM implementations.

a) OpenSSL

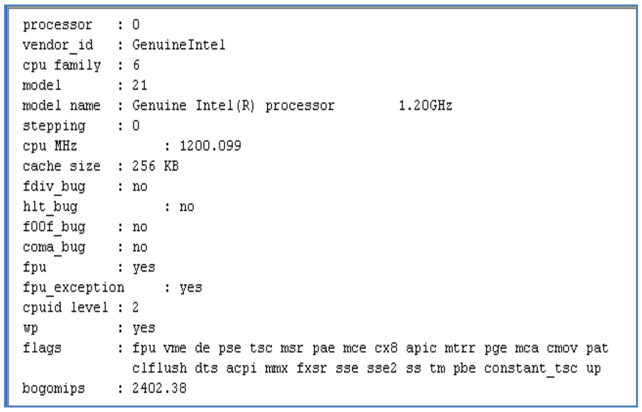
OpenSSL is an open source implementation of the Secure Socket Layer (SSL) and Transport Layer Security (TLS) protocols with the libcrypto library being the main component which implements a wide range of cryptographic algorithms used in various Internet standards. Libcrypto library has the concept of engines to allow other implementations to be plugged in, including hardware based accelerators. One of the most useful features is the ability to factor out processing intensive operations to specialist hardware through an 'engine' interface. It is through this engine subsystem that Costigan et al. [19] accelerate SSL by using the Cell (Cell Broadband Engine by IBM) SPU's (synergistic processor units) vector processing capabilities. Wrappers allowing the use of the OpenSSL library in a variety of computer languages are available. Versions are available for most Unix-like operating systems (including Solaris, Linux, Mac OS X and the four open source BSD Operating Systems), OpenVMS and Microsoft Windows. IBM provides a port for the System i (OS/400) [10].

b) Hardware Acceleration for OpenSSL

The Intel Document [11] describes how to patch OpenSSL to use the OpenBSD/FreeBSD Cryptographic Framework (OCF) engine on Linux. OCF is a service virtualization layer implemented inside the kernel that provides uniform access to accelerator functionality by hiding card-specific details behind a carefully-designed API [18]. OCF also includes a user-space library which allows the kernel driver to be accessed from user space via the /dev/crypto device. Keromytis et al. [18] have shown that the OCF is extremely efficient in utilizing cryptographic accelerator functionality, attaining 95% of the theoretical peak device performance, and over 800 Mbit/sec aggregate throughput using 3DES, though not specifically on Tolapai. Tolapai provides a "shim" or plugin for OCF to allow users of the OCF API in the kernel or user space to be offloaded to the integrated crypto accelerator. The OCF driver enables OpenSSL to accelerate cryptographic operations using the integrated cryptographic accelerator. The acceleration is handled by the CRYPTODEV engine, provided by OCF in the form of a plug-in to libcrypto.

IV. EXPERIMENTAL SETUP

We have two Intel EP80579 Development Boards at our disposal and they were assembled as per the instructions on the Intel user guide for EP80579 [12]. Fig. 3 and Fig. 4 depict the system configuration and a setup for experiments in our lab. RedHat Linux kernel was installed on these systems along with software drivers and kernel modules for the QuickAssist Technology provided by Intel. A software implementation of the Intel® QuickAssist Technology cryptographic API uses the integrated crypto accelerator of Tolapai. Both of these Tolapai boards were setup for DKIM benchmarking. One was used to map the acceleration provided by the hardware acceleration patched for OpenSSL as specified in Sec. III.C.b by utilizing the QuickAssist technology features whereas the other was used to setup the software counterpart of the application (for comparison purposes).



```
processor   : 0
vendor_id   : GenuineIntel
cpu family  : 6
model       : 21
model name  : Genuine Intel(R) processor        1.20GHz
stepping    : 0
cpu MHz           : 1200.099
cache size  : 256 KB
fdiv_bug    : no
hlt_bug           : no
f00f_bug    : no
coma_bug    : no
fpu         : yes
fpu_exception     : yes
cpuid level : 2
wp          : yes
flags       : fpu vme de pse tsc msr pae mce cx8 apic mtrr pge mca cmov pat
              clflush dts acpi mmx fxsr sse sse2 ss tm pbe constant_tsc up
bogomips    : 2402.38
```

Fig. 3: Tolapai system configuration used for testing

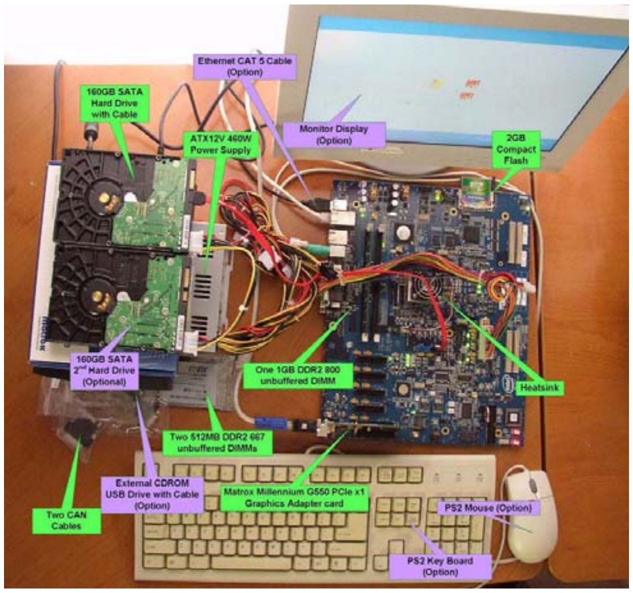

Ethernet CAT 5 Cable (Option)
160GB SATA Hard Drive with Cable
ATX12V 460W Power Supply
Monitor Display (Option)
2GB Compact Flash
160GB SATA 2nd Hard Drive (Optional)
One 1GB DDR2 800 unbuffered DIMM
Heatsink
Two 512MB DDR2 667 unbuffered DIMMs
External CDROM USB Drive with Cable (Option)
Matrox Millennium G550 PCIe x1 Graphics Adapter card
Two CAN Cables
PS2 Mouse (Option)
PS2 Key Board (Option)

Fig. 4: Intel EP80579 Lab setup [12]

A. BENCHMARKING

The details of the experiment are summarized in Table 1. The table shows the input, output, algorithms used and the number of runs used for averaging the results.

Table 1: Experiment Details

SOFTWARE	OpenSSL 0.9.8e
INPUT	using the OpenSSL provided benchmarking suite "speed" utility. - openssl speed <DKIM specific core Algorithm>
ALGORITHMS	SHA1, SHA256, RSA signing – verification for 1024 and 2048 bits
OUTPUT	profiling the performance of the core DKIM algorithms using OpenSSL crypto library using the setup described above
RUNS	100 (first 5 are reported)

a) OPENSSL'S BUILT-IN BENCHMARK RESULTS FOR DKIM

Fig. 5, which maps increasing block sizes against thousands of bytes on which the SHA1digest is applied per second, shows that as block size increases for SHA1 the Hardware version shows substantially improved performance. The performance of the hardware version is actually lower than software for smaller buffer sizes as the stack is non-optimal (i.e., libcrypto to OCF engine layer to OCF cryptodev, then into the kernel with buffer copying, then through OCF, OCF shim, then the Quick Assist library – and all the way back up again). A more optimal stack is possible, but has not been implemented at this time. Fig. 6 shows a sample output of the OpenSSL speed test with its various timelines for SHA1. The current OCF patch (OCF shim implemented for Tolapai nor the OpenSSL engine layer patch) doesn't provide hardware acceleration for SHA256 and hence there isn't any significant difference in performance between the software and hardware versions of DKIM for SHA256. Figures 7 through 10 show that moving the main functionality to the hardware produces a substantial performance boost in that we can RSA-sign and verify significantly more number of blocks per second.

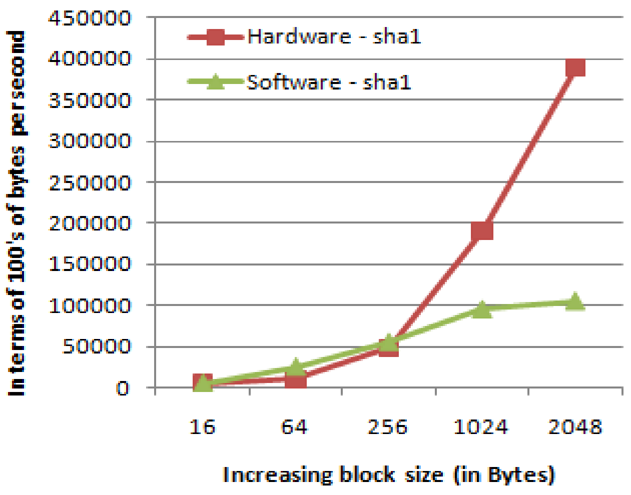
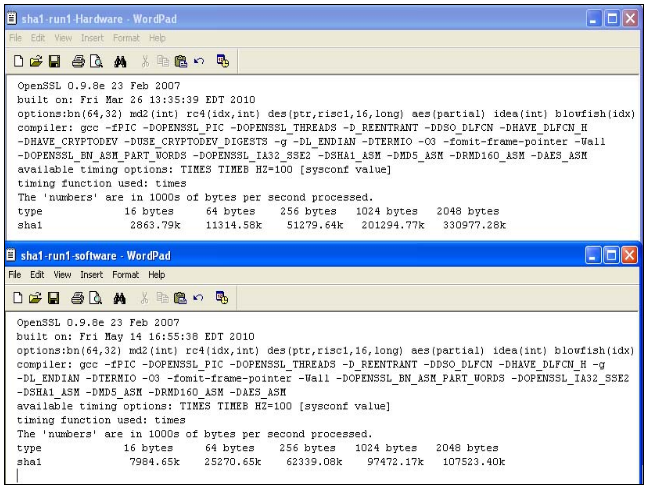


Fig. 5: OpenSSL SHA1 profiling


sha1-run1-Hardware - WordPad
File Edit View Insert Format Help

```
OpenSSL 0.9.8e 23 Feb 2007
built on: Fri Mar 26 13:35:39 EDT 2010
options:bn(64,32) md2(int) rc4(idx,int) des(ptr,risc1,16,long) aes(partial) idea(int) blowfish(idx)
compiler: gcc -fPIC -DOPENSSL_PIC -DOPENSSL_THREADS -D_REENTRANT -DDSO_DLFCN -DHAVE_DLFCN_H
-DHAVE_CRYPTODEV -DUSE_CRYPTODEV_DIGESTS -g -DL_ENDIAN -DTERMIO -O3 -fomit-frame-pointer -Wall
-DOPENSSL_BN_ASM_PART_WORDS -DOPENSSL_IA32_SSE2 -DSHA1_ASM -DMD5_ASM -DRMD160_ASM -DAES_ASM
available timing options: TIMES TIMEB HZ=100 [sysconf value]
timing function used: times
The 'numbers' are in 1000s of bytes per second processed.
type             16 bytes     64 bytes    256 bytes   1024 bytes   2048 bytes
sha1              2863.79k    11314.58k    51279.64k   201294.77k   330977.28k
```

sha1-run1-software - WordPad
File Edit View Insert Format Help

```
OpenSSL 0.9.8e 23 Feb 2007
built on: Fri May 14 16:55:38 EDT 2010
options:bn(64,32) md2(int) rc4(idx,int) des(ptr,risc1,16,long) aes(partial) idea(int) blowfish(idx)
compiler: gcc -fPIC -DOPENSSL_PIC -DOPENSSL_THREADS -D_REENTRANT -DDSO_DLFCN -DHAVE_DLFCN_H -g
-DL_ENDIAN -DTERMIO -O3 -fomit-frame-pointer -Wall -DOPENSSL_BN_ASM_PART_WORDS -DOPENSSL_IA32_SSE2
-DSHA1_ASM -DMD5_ASM -DRMD160_ASM -DAES_ASM
available timing options: TIMES TIMEB HZ=100 [sysconf value]
timing function used: times
The 'numbers' are in 1000s of bytes per second processed.
type             16 bytes     64 bytes    256 bytes   1024 bytes   2048 bytes
sha1              7984.65k    25270.65k    62339.08k    97472.17k   107523.40k
```

Fig. 6: Sample SHA1 Run Output

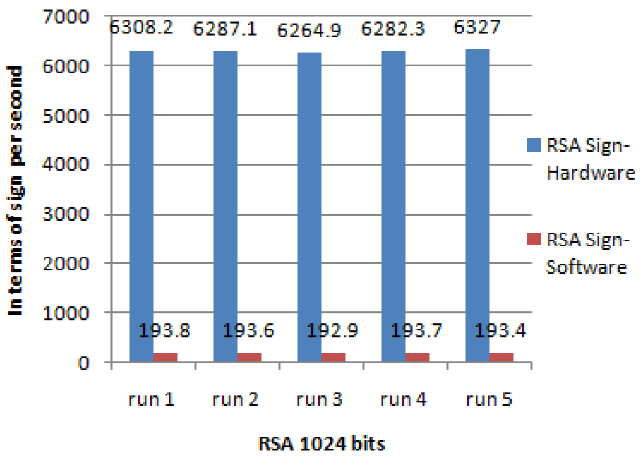


Fig. 7: OpenSSL RSA signing - 1024 bits profiling

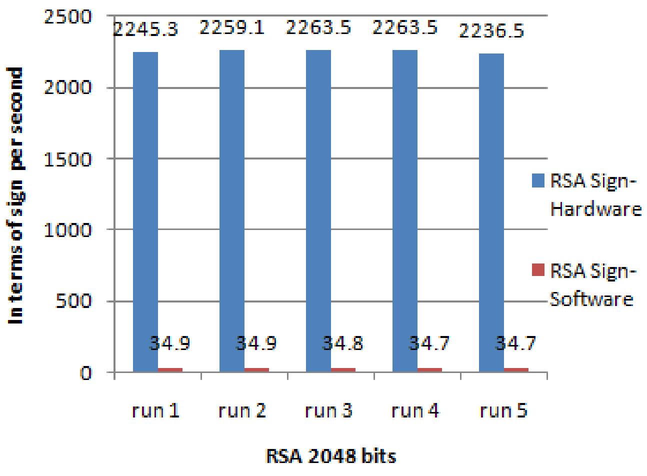


Fig. 8: OpenSSL RSA signing - 2048 bits profiling

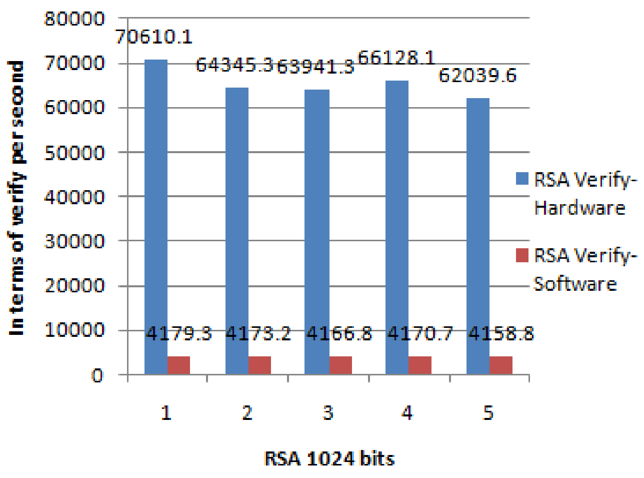


Fig. 9: OpenSSL RSA verification - 1024 bits profiling

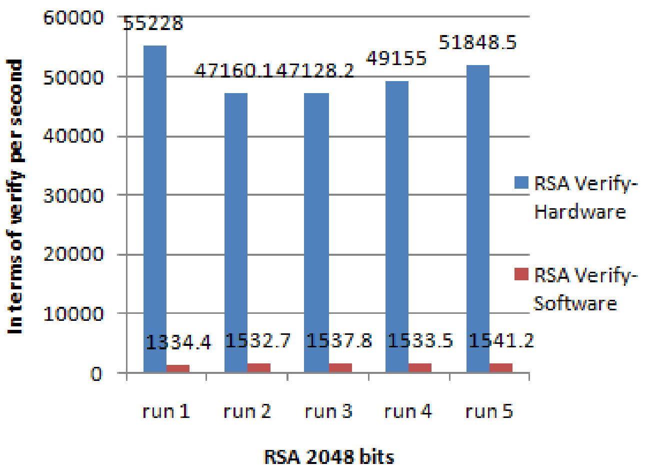
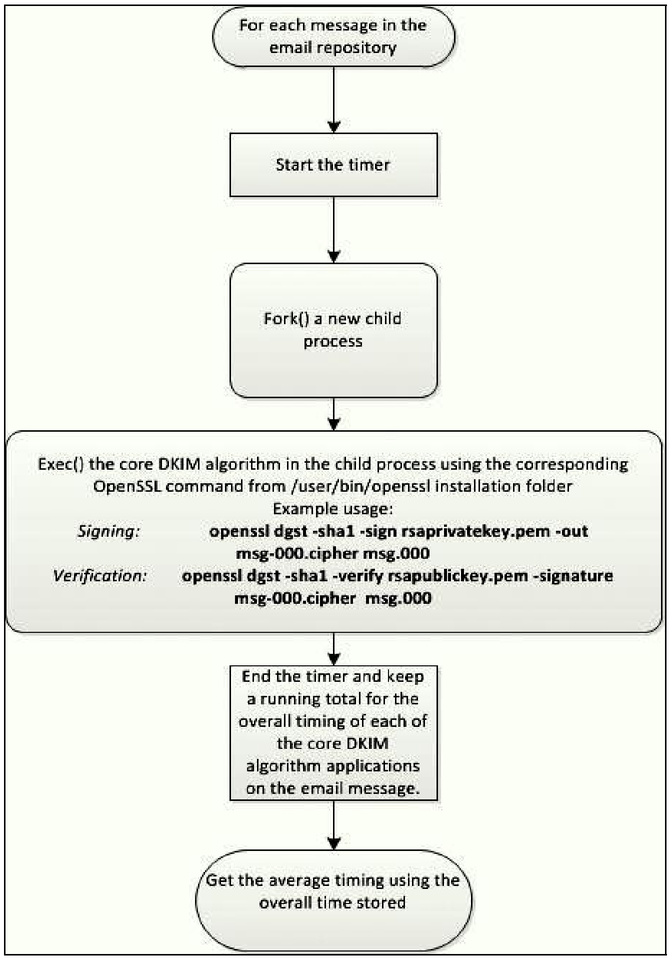


Fig. 10: OpenSSL RSA verification - 2048 bits profiling

b) END-TO-END IMPLEMENTATION BENCHMARK RESULTS FOR DKIM

A dataset from an earlier research project [13] was used for the emails, which consists of 1197 messages with average size of 8.8KB, median of 7KB and the majority of messages in the range of 1-20 KB. There were no messages larger than 180KB. A code snippet in C using the threading model was developed to map the timings as compared to a shell script program to achieve the same result for the purposes of getting the timing in milli-seconds which isn't possible via shell scripting.

The code was written for calculating the average time of applying the core modules of DKIM (which use OpenSSL for the algorithms) on the email repository to determine the end-to-end performance enhancements, the flow of which is as shown in Fig. 11. This implementation doesn't include publishing and retrieval of the public key from the DNS record along with additions of the message header tags related to DKIM. The same code was run on both Tolapai setups – one utilizing the hardware features and the other not utilizing the hardware, and the results are noted in tables 2 through 5. The speedups are apparent from the tables.


For each message in the email repository
Start the timer
Fork() a new child process
Exec() the core DKIM algorithm in the child process using the corresponding OpenSSL command from /user/bin/openssl installation folder
Example usage:
Signing:
Verification:
End the timer and keep a running total for the overall timing of each of the core DKIM algorithm applications on the email message.
Get the average timing using the overall time stored

```
Signing:       openssl dgst -sha1 -sign rsaprivatekey.pem -out
               msg-000.cipher msg.000
Verification:  openssl dgst -sha1 -verify rsapublickey.pem -signature
               msg-000.cipher  msg.000
```

Fig. 11: End-to-end implementation code flow

Table 2: RSA – 1024 bits with SHA1

	Hardware (Avg Time - ms)	Software (Avg Time - ms)
Signing	6.072024	9.946899
Verification	6.469257	6.677209

Table 3: RSA – 1024 bits with SHA256

	Hardware (Avg Time - ms)	Software (Avg Time - ms)
Signing	6.417706	9.999939
Verification	6.651458	6.932569

Table 4: RSA – 2048 bits with SHA1

	Hardware (Avg Time - ms)	Software (Avg Time - ms)
Signing	6.160404	13.968011
Verification	6.502682	7.183801

Table 5: RSA – 2048 bits with SHA256

	Hardware (Avg Time - ms)	Software (Avg Time - ms)
Signing	6.189453	13.984166
Verification	6.621802	7.185308

V. DISCUSSION OF RESULTS

Though the results of Section IV.A.a are shown only for the first five runs, the experiment was run 100 times and averaged so that any noisy timing measurements are eliminated. Our experiments show a 30x and 15x boost in performance for 1024-bits RSA signing and verification respectively, as a result of moving to hardware. These figures are twice as good for 2048-bits RSA signing and verification. A summary of the results of our end-to-end benchmark experiments in Section IV.A.b is presented in Table 6. The results in Section IV.A.a show a much higher relative performance difference between the hardware and software based implementations when compared with the results of Section IV.A.b. This is because the former experiments use the optimal inbuilt OpenSSL commands and the latter end-to-end performance experiments include multiple core DKIM algorithms, some of which are inherently slower. Overall, the data shows that performance for DKIM algorithm implementation and DKIM milters can be significantly increased by using the Tolapai processor. Our approach not only achieves better performance but at the same time doesn't require a dedicated hardware co-processor or chip.

Table 6: End-to-end gains by hardware implementation

Algorithm used in DKIM	**Signing/Verification**	**%Gain w.r.t Software**
RSA – 1024 bits with SHA1	Signing	63.82%
RSA – 1024 bits with SHA1	Verification	3.21%
RSA – 1024 bits with SHA256	Signing	55.82%
RSA – 1024 bits with SHA256	Verification	4.23%
RSA – 2048 bits with SHA1	Signing	126.74%
RSA – 2048 bits with SHA1	Verification	10.47%
RSA – 2048 bits with SHA256	Signing	125.94%
RSA – 2048 bits with SHA256	Verification	8.51%

Experimental results in Table 6 show that an overall acceleration of 49.84% is achieved by transparently migrating the DKIM functionalities implemented using OpenSSL library to hardware.

Being a commercial off-the-shelf (COTS) processor, no additional chip is required to achieve this acceleration. The interface to hardware through APIs that are provided by Intel, make the acceleration completely transparent to the user. The implementation of DKIM milter is still flexible, thus, overcoming shortcomings of any dedicated hardware modules. Other security applications can make use of the accelerating capabilities of Tolapai without additional costs. Our conclusion is that DKIM milters utilizing the Tolapai processor features will perform better in applying the DKIM algorithms on email messages and could detect spam and phishing emails in real-time even when the traffic volume is significantly high.

VI. FUTURE WORK

Our future work would consist of improving the OpenSSL OCF driver for SHA-256 performance acceleration required for DKIM. We plan to move the full end-to-end implementation of DKIM milter at an MTA to the hardware and profile the performance. We also plan on utilizing Tolapai to implement other security applications which will be more efficient in performance than their software counterparts. We have already done some preliminary research in this area by moving certain functions of the Naïve Bayesian spam filters to Intel EP80579 hardware [1]. Moving more functions such as "tokenizing" which is a major time consuming function in spam filtering will be a useful future work. Finally, we will also look into exploiting the acceleration features of other (upcoming) SoC processors such as the Stellarton [20] to achieve higher performances for security applications (such as secure email) on embedded processor platforms.

ACKNOWLEDGEMENTS

This research has been done in part through a grant from Intel Corporation. The authors would like to thank Pranil Gupta and Ajay Nagrale for the useful discussions during the course of this research.

REFERENCES

[1] Pranil Gupta, Ajay Nagrale and Shambhu Upadhyaya, "Accelerating Techniques for Rapid Mitigation of Phishing and Spam Emails." *Workshop on Embedded Systems and Network Security, in conjunction with IEEE SRDS 2009, Sept. 2009*. Available: http://www.cse.buffalo.edu/srds2009/escs2009_submission_Gupta.pdf.

[2] Schatzmann, Martin Burkhart and Thrasyvoulos Spyropoulos, "Flow-level Characteristics of Spam and Ham." *TIK-Report No. 291, Computer Engineering and Networks Laboratory, ETH Zurich, Switzerland, 29.* August 2008.

[3] Product Brief - Intel® EP80579 Integrated Processor with Intel® QuickAssist Technology Embedded Computing [Online]. Available: http://download.intel.com/design/intarch/ep80579/319944.pdf

[4] E. Allman, J. Callas, M. Delany, M. Libbey, J. Fenton and M. Thomas, "DomainKeys Identified Mail (DKIM)", *Internet Engineering Task Force, RFC 4871,* May, 2007.

[5] E. Allman, M. Delany and J. Fenton, "DKIM Sender Signing Practices", *Internet Draft*, http://www.ietf.org/internetdrafts/draft-allman-dkim-ssp-09.pdf, Feb, 2009.

[6] Barry Leiba and Jim Fenton, "DomainKeys Identified Mail (DKIM): Using Digital Signatures for Domain Verification." *CEAS 2007 – Fourth Conference on Email and Anti-Spam, August 2-3, 2007,* Mountain View, California, Available: http://www.ceas.cc/2007/papers/paper-78.pdf, http://domino.research.ibm.com/comm/research_people.nsf/pages/leiba.pubs.html/$FILE/dkim_rc23995.pdf.

[7] Stephen Farrell, "DomainKeys Identified Mail Demonstrates Good Reasons to Re-invent the Wheel." *Public key infrastructure: Third European PKI workshop: theory and practice, EuroPKI 2006*, Turin, Italy, June 2006, edited by Andrea S. Atzeni, Antonio Lioy. http://books.google.com/books?hl=en&lr=&id=tvydhUrrDUIC&oi=fnd&pg=PA145&ots=0_uEi4liP5&sig=g9VXohpq4WpqTjDRThPKkQ1Xpxk#v=onepage&q&f=false , visited Dec 25th, 2010.

[8] M. Libbey, "DKIM fights phishing and e-mail forgery", http://www.networkworld.com/news/tech/2005/080805techupdate.html, visited July 26, 2010.

[9] http://www.sendmail.com/sm/wp/dkim/, visited July 26, 2010.

[10] Wikipedia, OpenSSL, (electronic) – The free encyclopedia. Available [Online]. http://en.wikipedia.org/wiki/OpenSSL.

[11] Installing Accelerated OpenSSL (OCF) and Apache* on Linux* — For use with Intel® EP80579 Software for Security Applications on Intel® QuickAssist Technology, Application Notes, *September 2008.*

[12] Intel® EP80579 Integrated Processor with Intel® QuickAssist Technology Development Kit User Guide [Online]. Available: http://download.intel.com/design/intarch/ep80579/320067.pdf

[13] Madhusudhanan Chandrasekaran, Krishnan Narayanan, Shambhu Upadhyaya. "Phishing Email Detection based on Structural Properties", *NYS Cyber Security Conference*, Albany, NY, June 2006.

[14] Yan Luo, "Workload characterization of spam email filtering systems", *International Journal of Network Security & Its Application (IJNSA)*, Vol. 2, No. 1, January 2010.

[15] http://origindownload.advantech.com//productFile/1-3LI2F1/Manual-FWA-3240-1st_Ed.pdf, visited Dec 25th, 2010.

[16] http://origindownload.advantech.com/ProductFile/1-FTBFVT/FWA-3240_DS(10.6.17).pdf, visited Dec 25th, 2010.

[17] Wikipedia, Milter, (electronic) – The free encyclopedia. Available [Online]. http://en.wikipedia.org/wiki/Milter., visited Dec 25th, 2010.

[18] Angelos D. Keromytis, Jason L. Wright and Theo de Raadt. *OpenBSD Project and Columbia University*. http://www.openbsd.org/papers/ocf.pdf, visited Dec 25th, 2010.

[19] Costigan, N., Scott, M.: Accelerating SSL using the vector processors in IBM's Cell broadband engine for Sony Playstation, SPEED 2007 Workshop, 2007.

[20] http://www.intel.com/newsroom/kits/idf/2010_fall/gallery_keynotes.htm?wapkw=%28stellarton%29. Visited June 9, 2011.