Crypto Accelerators for Power-Efficient and Real-Time on-Chip Implementation of Secure Algorithms

Luca Baldanzi
Dept. of Information Engineering
University of Pisa
Pisa, Italy
luca.baldanzi@ing.unipi.it

Luca Crocetti
Dept. of Information Engineering
University of Pisa
Pisa, Italy
luca.crocetti@phd.unipi.it

Stefano Di Matteo
Dept. of Information Engineering
University of Pisa
Pisa, Italy
stefano.dimatteo@dii.unipi.it

Luca Fanucci
Dept. of Information Engineering
University of Pisa
Pisa, Italy
luca.fanucci@unipi.it

Sergio Saponara
Dept. of Information Engineering
University of Pisa
Pisa, Italy
sergio.saponara@unipi.it

Patrice Hameau
Senior Security Architect
Prove & Run
Paris, France
patrice.hameau@provenrun.com

***Abstract*—The demand for data exchange is ever growing. Internet of Things (IoT), industry 4.0, smart city and next-generation interconnected vehicles are some examples of scenarios in which a high volume of nodes share data across networks. Hence, the data protection plays a fundamental aspect to avoid disclosure or manipulation of sensitive information and disruption of services, particularly in safety critical applications. On the other hand, also the compute power at disposal of possible attackers and hackers is growing, and next-future post-quantum capabilities will require the usage of longer keys, certificates and digital signatures, to preserve the security level offered by cryptographic functions. This will affect not only the amount of exchange data, but also the computational resources to secure data, increasing processing time, latencies and power consumption, and lowering data rates. In this work, we investigate different implementation strategies to overcome such performance limitations. This work provides a comparison among pure software approach (both on 32b and 64b processors) and hardware-based solutions we developed for FPGA and ASIC System-on-Chip platforms, for the most common symmetric-key and public-key cryptographic algorithms. The proposed hardware accelerators feature one order of magnitude higher throughput (and lower latency) and more than two orders lower power consumption than their software counterparts. A highly configurable cryptographic suite is proposed that can be customized according to the application requirements and thus able to increase as much as possible the efficiency in terms of energy per enciphered bits per second.**

Keywords—Cryptographic hardware accelerators, energy efficiency, ARM 32-bit, ARM 64-bit

I. INTRODUCTION

The amount of data exchanged in modern electronic systems is constantly growing. Some examples of these systems are Internet of Things (IoT), interconnected vehicles, Industry 4.0, cloud-based services. From an operating point of view, the high networking level of electronic systems guarantees a higher level of quality service for end users, but it introduces potentially vulnerabilities in terms of cyber security. A typical example that explains this concept is the autonomous driving application: in the next years cars will be able to communicate together in order to provide to drivers and passengers a better driving experience and enhanced services (e.g. data exchange for traffic information, road obstacle detection, etc.). The V2X (Vehicle to Everything) capability introduces an enlargement of the attack surface that a malicious entity could exploit for data stealing, service interruption and so on. For these reasons, interconnected systems require to be equipped with appropriate cryptographic algorithms, in order to protect the communication against cyberattacks. In addition, these algorithms shall meet stringent requirements in terms of throughput, power consumption, and usage security policy.

In [1] the authors analysed and characterized the performance costs (in terms of power consumption and execution times) of software solutions for cryptographic services, targeting the ARM Cortex-M processors family. As conclusion of that work, the authors stated that despite Cortex-M processors were capable of running standard cryptographic algorithms such Advanced Encryption Standard (AES) and Secure Hash Algorithm (SHA), without excessive power consumption, the use of a hardware cryptographic module or an algorithm optimisation could be required for that processors in case of public-key cryptography: “as the resulting execution times for Elliptic Curve Digital Signature Algorithm (ECDSA) were sufficiently long as to increase the probability of missed deadlines in a hard real-time application” [1]. For this reason, in this research activity we used a Cortex-A53 processor by ARM, which includes the ARMv8 Cryptographic Extension module, providing instructions for accelerating the execution of AES Decryption, AES Encryption, SHA1, SHA224 and SHA256 algorithms [2]. In addition, we developed hardware modules for accelerating cryptographic functions, in order to characterize and compare both hardware (HW) and software (SW) implementations in terms of power consumption, throughput/data rate and energy per bit.

The remainder of this work is organized as it follows: Section II describes the methods, the equipment and the tools used to characterize the performance of both software and hardware solutions. Section III reports the acquired data in term of throughput, latency, power consumption and energy per bit for the AES, SHA256 and ECDSA algorithms. Accordingly, Section IV gives the conclusions of this work.

II. METHODS AND MATERIAL

As mentioned in Section I, in the SW case we targeted an ARM Cortex-A53 processor, which is a 64-bit processor featuring the ARMv8 architecture. As ARM does not physically produce silicon, there are several vendors from whom the processors can be purchased. As a result, variations in processing speed and available peripherals can occur. For the purposes of this research, we used the 4-Core Broadcom BCM2837B0 chip running @ 1.4 GHz in 40 nm technology. The Broadcom chip is mounted on Raspberry Pi 3B+ board

[3], and is representative of multicore Cortex-A53 performance, embedded also in automotive application processor such as NXP-S32V [4].

To perform the cryptographic functions in SW, we employed the OpenSSL library (version 1.1.1c, [5]), which is a robust, commercial-grade and full-featured toolkit for TLS (Transport Layer Security) and SSL (Secure Sockets Layer) protocols [6]. It provides a full-range of symmetric-key and public-key algorithms and a built-in self-test routine named *speed* [7]. Such routine allows selecting an algorithm to be executed, and then it executes the specified algorithm for a fixed amount of time (typically 3 or 10 seconds); at the end of the process, the routine reports the number of times such algorithm has been executed. This let us characterize the SW solution in terms of throughput and latency. Concerning the power consumption, we used an approach similar to the one employed in [1], [8-10]. We fed the board hosting the BCM2837B0 chip with a constant voltage of 5 V provided by a bench power supply (Keysight E3631A Triple Output DC Power Supply), and we acquired the absorbed current by the board, by means of a digital multimeter (Keysight 34461A Digital Multimeter), while executing the *speed* routine for several algorithms. Thus, we post-processed the acquired data to retrieve the average dynamic power consumption of the 4-Core Cortex-A53 chip, excluding the power consumption contribution of the other board components, in order to make a fair comparison with the our designed HW crypto accelerator counterpart.

In the HW case, we developed dedicated hardware modules for accelerating cryptographic functions. The modules have been designed by describing them in SystemVerilog HDL language and then synthetized with the Synopsys Design Compiler tool and using the open-source standard-cell library provided by the Silvaco 45nm Open Cell Library [11]. In order to support a wide range of cryptographic functions able cover the typical and most diffused algorithms employed in symmetric-key and public-key cryptographic applications, we developed the following hardware accelerators:

- an AES core, featuring both encryption and decryption and support to both 128-bit and 256-bit cipher keys compliant with standard [12][13][14];
- a SHA engine, supporting all the possible digest sizes (i.e. 224, 256, 384 and 512 bits) compliant with standard [15][13][14];
- an ECDSA engine, supporting the NIST P-256 and NIST P-521 elliptic curves compliant with standard [16][13].

The hardware modules have been developed by University of Pisa for the purpose to be integrated in exascale capable processors developed by the European Processor Initiative (EPI) consortium [17].

On the power consumption side, we extracted the required data by running simulations of the synthetized modules with Synopsys PrimeTime tool and linking to it the same standard-cell library used for synthesis, in order to gather the power consumption information included in that library. On the throughput, data rate and latencies side, once determined the maximum frequency our modules were able to support by synthesis, we have been able to compute mathematically such data basing only on the modules architecture.

Anyway, to make a fair comparison with the SW counterpart, we considered also the latencies due to a real-case application. In other words, in the computation we included also the time a processor integrating our modules should spent in order to use them. The typical approach is to provide hardware accelerators with a bus interface and connect them to the system bus that the processor accesses to, as peripherals. In order to gain advantage from the maximum speed achievable by our modules, on one hand we assumed the utilization of an AXI4-DMA interface [18] to stream data from the system memory to our modules and vice versa. Moreover, we assumed to apply the same clock frequency to both the AXI bus and our modules, in order to reduce overhead due to synchronization mechanisms. On the other hand, we fixed the clock frequency to 300 MHz, in order to reduce the power consumption of hardware accelerators. Such frequency is largely supported by both the AXI4 bus [18] and our modules, being the maximum achievable frequency 560 MHz, 1.1 GHz and 325 MHz, respectively, for the AES core, the SHA engine and the ECDSA engine, by results of synthesis on the 45 nm standard-cell technology. Thus, the maximum bandwidth reachable using the AXI4-DMA is about 9.6 Gbps and 3.6 Gbps for MM2S and S2MM channels respectively [18], when using a bus clock of 300 MHz. The MM2S (memory-mapped to stream) transfers apply to data writing to peripheral (i.e. our hardware accelerator), while the S2MM (stream to memory-mapped) transfers apply to data reading from peripheral. To be noted that the target technology for the first tape-out of EPI project will be on 7 nm standard-cell technology, and from preliminary synthesis result there is an increase of the maximum clock frequency by a factor 5 with respect to the above 45 nm data.

III. RESULTS

For the purposes of this research activity, we analysed the following cryptographic functions:

- AES-ECB-256 (encryption), i.e. the encryption algorithm of the ECB mode [19] of the AES cipher, with 256-bit cipher key [12][13][14];
- SHA256, i.e. the SHA algorithm when generating 256-bit digests [15][13][14];
- ECDSA signature generation on NIST P-521 curve [16][13].

Table I reports the performance data for the AES-ECB-256 software implementation, and Fig.1 shows the dynamic power consumption of the BCM2837B0 chip during the execution of AES-ECB-256 algorithm. In Table I, Table II and Table III, *Exec. time* column reports the execution time, *D* column reports the amount of processed data in Mbit, *TH* column reports the throughput @ 1.4 GHz, the *P* column reports the average dynamic power consumption, and the *E* column reports the energy per bit.

TABLE I. Performance data for AES-ECB-256 software implementation.

Number of core	Exec. time (s)	D (Mb)	TH (Mbps)	P (mW)	E (mJ/Mb)
1	3	917.4	305.80	300	0.98
2	3	1812.8	604.27	600	0.99
4	3	3628	1209.33	1300	1.07

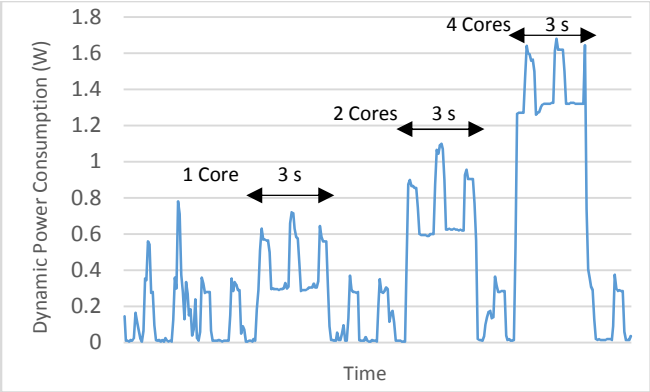

Figure 1. Dynamic power acquired during execution of AES-ECB-256 algorithm on ARM Cortex-A53 processor, using 1, 2 and 4 cores.

Similarly, Table II and Table III report data characterizing the software implementations of SHA256 and the ECDSA, respectively, as well as Fig. 2 and Fig. 3 show the dynamic power consumption during the execution of such algorithms. To be noted that data in Fig. 1, Fig. 2 and Fig. 3 have been post-processed to extract the average dynamic power consumption for the different cases of number of cores employed during the execution of the algorithm.

TABLE II. Performance data for SHA256 software implementation.

Number of core	Exec. time (s)	D (Mb)	TH (Mbps)	P (mW)	E (mJ/Mb)
1	3	337.9	113,01	310	2.74
2	3	664.9	221,63	650	2.93
4	3	1213.9	404,63	1380	3.14

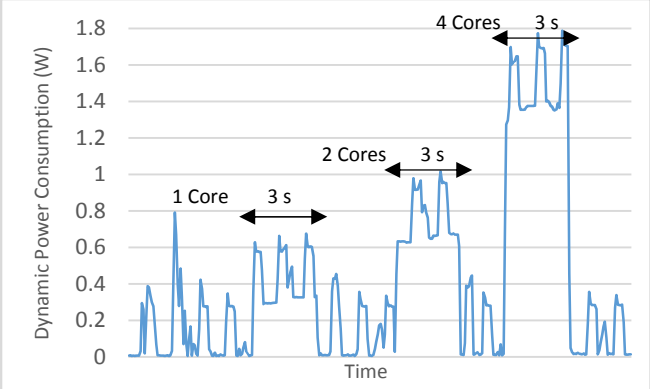

Figure 2. Dynamic power acquired during execution of SHA256 algorithm on ARM Cortex-A53 processor, using 1, 2 and 4 cores.

TABLE III. Performance data for ECDSA software implementation.

Number of core	Exec. time (s)	D (Op)	TH (Op/s)	P (mW)	E (mJ/Op)
1	10	282.4	28.24	310	10.98
2	10	560	56	620	11.07
4	10	1085	108.5	1330	12.26

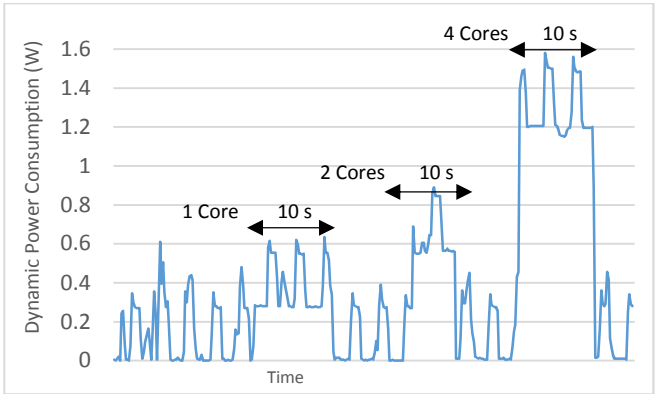

Figure 3. Dynamic power measured during execution of ECDSA over NIST P-521 curve algorithm using 1, 2 and 4 cores.

Concerning the HW implementation, Table IV shows data characterizing the developed cryptographic accelerators, for a frequency of 300 MHz. As specified in Section II, in this case we included also the additional latency affecting throughput, due to software access to our modules in a real-case scenario. Hence, assuming the bandwidth achieved by the AXI4-DMA at the frequency of 300 MHz for both MM2S and S2MM transfers, the additional latency to be taken in account is:

- 48.89 ns, for AES-ECB-256;
- 124.44 ns, for SHA256;
- 316.11 ns, for ECDSA signature generation on NIST P-521 curve.

Accordingly, the throughput of the hardware accelerators can be calculated by:

$$TH = \frac{Q}{L*T_{clk}+AL} \quad (1)$$

Referring to Eq. (1), T_{clk} is the clock period of the clock frequency of 300 MHz, i.e. 5 ns, and L is the module latency, expressed in clock cycles, and AL is the additional latency due to software access and reported in the list above, and expressed in seconds. This last one parameter can be computed by dividing the bit width of the amount of input (or output) data required (or generated) by the hardware module, by the AXI4-DMA bandwidth, in case of MM2S channel (or S2MM channel). Q strongly depends on the cryptographic function and it is:

- Q = 128 (bits), in case of AES-ECB-256, because ECB mode of AES requires one 128-bit input data block and generates one 128-bit output block, for every 128-bit block composing a message;
- Q = 512 (bits), in case of SHA256, because such algorithm requires to parse all 512-bit blocks composing a message, to generate the corresponding 256-bit digest, thus, in this case, the throughput refers to the input data rate supported by the input interface;
- Q = 1 (operation), in case of ECDSA signature generation algorithm, because it is a single-shot operation and provides a set of output data (i.e. the digital signature), after providing the required input data.

TABLE IV. Performance data of hardware cryptographic accelerators. The TH column reports the throughput, the P column reports the average dynamic power consumption, and the E column reports the energy per bit (or operation), according to the algorithm in the rows. All data refer to the synthesis on 45 nm standard-cell technology at the frequency of 300 MHz.

Cryptographic algorithm	TH	P (mW)	E
AES-ECB-256	1339.53 Mbps	18	0.00656 mJ/Mb
SHA256	1515.79 Mbps	8.91	0.00371 mJ/Mb
ECDSA-P521	910.60 Op/s	61.8	0.06785 mJ/Op

Table V shows the comparison among software and hardware implementations of the cryptographic algorithms we analyzed in this work, using as efficiency factor a figure of merit we computed as

$$FoM = \frac{TH}{E}$$

for which, *TH* and *E* are, respectively, the throughput and the energy per bit (or per operation) reported in Table I, Table II, Table III and Table IV for SW on 4-Core Cortex-A53 @ 1.4 GHz (40 nm) and our cryptographic HW accelerators @ 300 MHz (on 45 nm technology).

IV. CONCLUSIONS

In this work, we evaluated the performance of most common and diffused cryptographic algorithms comparing different implementations: pure SW solutions based on 64-bit processor (4-Core ARM Cortex-A53), and HW accelerators specifically designed and assumed to be used as peripherals of a GPP (General Purpose Processor), by means of an AXI4-DMA interface. In case of SW, we characterized the algorithms in three different conditions, varying the number of cores running the cryptographic applications (respectively 1, 2 or 4 cores). In case of HW, data refer to synthesis of digital modules on a 45 nm standard-cell technology. As consequence of results reported in Section III, it can be stated that in all the cases HW solutions are better than their SW counterparts in terms of throughput and power consumption, with an average *FoM* improvement of three orders of magnitude (refer to Table V). Anyway, pure SW solutions can be suitable in applications where the main target is a large system flexibility, without consider throughput and power consumption as benchmark metrics.

The throughput of our HW accelerators measured in this work is not the highest one that they can reach. For example, according to the application requirements, the HW AES can achieve up to 5 Gbps of throughput, in case of AES-ECB-256, at the maximum supported frequency of 560 MHz on 45 nm technology. A further x5 speed performance improvement is expected, in EPI target 7 nm technology, from preliminary synthesis results. In case of the most intensive algorithms, such as the ECDSA one, results prove that the HW accelerator solution is mandatory, since the relatively long execution times of SW ECDSA could be intolerable within a real-time application, or too expensive in terms of energy consumption for some markets (e.g. HPC or automotive).

As conclusion, even if using 64-bit multicore application processor, such as the Broadcom BCM2837B0 @ 1.4 GHz, the hardware-based solutions for cryptography are the only ones capable to meet stringent requirements in terms of real-time execution and power consumption for embedded IoT and automotive systems

Table V. Comparison among software and hardware implementations of cryptographic algorithms, according to Figure of Merit (FoM).

Algorithm	*FoM*			
	SW (1 core)	SW (2 cores)	SW (4 cores)	HW
AES-ECB-256	312.04	610.37	1130.21	204119.60
SHA256	41.24	75.64	128.86	408293.46
ECDSA P-521	2.57	5.06	8.85	20131.61

ACKNOWLEDGMENT

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826647.

REFERENCES

[1] L. P. I. Ledwaba G. P. Hancke H. S. Venter, and S. J. Isaac "Performance costs of software cryptography in securing new-generation internet of energy endpoint devices," IEEE Access vol. 6 pp. 9303-9323 Jan. 2018.

[2] Arm Limited. (2019). (Cortex-A53). Accessed: Jun. 5, 2019. Available: https://developer.arm.com/ip-products/processors/cortex-a/cortex-a53.

[3] Raspberry Pi. (Raspberry Pi 3 – Hardware – BCM2837B0). Accessed: May 28, 2019. Available: https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2837b0/README.md.

[4] NXP-S32 Automotive Processor and Microcontrollers. Available: https://www.nxp.com/products/processors-and-microcontrollers/arm-based-processors-and-mcus/s32-automotive-platform:S32

[5] OpenSSL. (Software library). Accessed: May 31, 2019. Available: https://www.openssl.org/source/openssl-1.1.1c.tar.gz.

[6] OpenSSL. (Main page). Accessed: May 30, 2019. Available: https://www.openssl.org/.

[7] OpenSSL. (Library manual). Accessed: May 31, 2019. Available: https://github.com/openssl/openssl/tree/master/doc/man1.

[8] C. C. Chang, S. Muftic, and D. J. Nagel, ''Measurement of energy costs of security in wireless sensor nodes,'' in Proc. 16th Int. Conf. Comput. Commun. Netw., Honolulu, Hi, USA, 2007, pp. 95–102.

[9] G. Guimaraes, E. Souto, D. Sadok, and J. Kelner, ''Evaluation of security mechanisms in wireless sensor networks,'' in Proc. Syst. Commun. (ICW'ICHSN'ICMCS'SENET'), Montreal, QC, USA, 2005, pp. 428–433.

[10] A. Trad, A. A. Bahattab, and S. B. Othman, ''Performance trade-offs of encryption algorithms for wireless sensor networks,'' in Proc. World Congr. Comput. Appl. Inf. Syst. (WCCAIS), Hammamet, Tunisia, 2014, pp. 1–6.

[11] Silvaco. (Open-source stand-cell library, 45nm OCL). Available: https://www.silvaco.com/products/nangate/FreePDK45_Open_Cell_Library/index.html

[12] "Advanced Encryption Standard (AES)," NIST Federal Information Publication Standard (FIPS) 197, Nov. 2001.

[13] "Référentiel Général de Sécurité," Règles et recommandations concernant le choix et le dimensionnement des mécanismes cryptographiques, Version 2.03, 2014.

[14] "Cryptographic Mechanisms: Recommendations and Key Lengths", BSI TR-02102-1, February 22, 2019.

[15] "Secure Hash Standard", NIST Federal Information Publication Standard (FIPS) 180-4, Mar. 2012.

[16] "Digital Signature Standard (DSS)," NIST Federal Information Publication Standard (FIPS) 186-4, Jul. 2013.

[17] European Processor Initiative (EPI). (Official website). Available: https://www.european-processor-initiative.eu/

[18] Xilinx. (AXI DMA – Guide). Available: https://www.xilinx.com/support/documentation/ip_documentation/axi_dma/v7_1/pg021_axi_dma.pdf

[19] "Recommendation for Block Cipher Modes of Operation", NIST Special Publication (SP) 800-38A, 2001.