Performance Evaluation of SHA-2 Standard vs. SHA-3 Finalists on Two Freescale Platforms

Pal-Stefan Murvay, Politehnica University of Timisoara, Timisoara, Romania

Bogdan Groza, Politehnica University of Timisoara, Timisoara, Romania

ABSTRACT

Embedded devices are ubiquitously involved in a large variety of security applications which heavily rely on the computation of hash functions. Roughly, two alternatives for speeding up computations co-exist in these resource constrained devices: parallel processing and hardware acceleration. Needles to say, multi-core devices are clearly the next step in embedded systems due to clear technological limitations on single processor frequency. Hardware accelerators are long known to be a cheaper approach for costly cryptographic functions. The authors analysis is focused on the five SHA-3 finalists which are also contrasted to the previous SHA-2 standard and to the widespread MD5. On the hardware side, the authors deploy their implementations on two platforms from Freescale: a S12X core equipped with an XGATE coprocessor and a Kinetis K60 core equipped with a crypto co-processor. These platforms differ significantly in terms of computational power, the first is based on a 16-bit Freescale proprietary architecture while the former relies on a more recent 32-bit Cortex core. The authors 'experimental results show mixed performances between the old standard and the new candidates. Some of the new candidates clearly outperform the old standard in terms of both computational speed and memory requirements while others do not. Bottom line, on the 16 bit platform BLAKE and Grøstl are the top performers while on the 32-bit platform Keccak, Blake and Skein give the best results.

Keywords: *Coprocessor, Hash Function, Parallelism, SHA-2, SHA-3*

INTRODUCTION AND MOTIVATION

Microcontrollers are ubiquitous devices associated with various security tasks, an increasingly notable aspect in the recent years due to their use in safety critical environments that are becoming more complex, e.g., control systems, industrial networks, vehicular technology, etc. Also, their connection to the outside world is unabridged at least due to the wide spread of friendly communication interfaces that become a good source for attacks, e.g., malicious code injection, etc. Regardless of the security objectives that need be assured, hash functions are an invariant choice in deploying security.

DOI: 10.4018/ijsse.2013100101

Several practical scenarios in which hash functions are involved can be imagined, e.g., software validation, embedded communications, etc. In particular firmware updates in embedded platforms (which require cryptographic hash functions for the protection of intellectual property, data integrity or non-repudiation) can directly benefit from performance improvements. Notably, digital signatures are employed to ensure that only an authentic firmware is programmed on a certain embedded device (Nilsson et al., 2008). Verifying signatures on a constrained embedded device can be a time consuming task especially as the size of the applications is continuously increasing (Petters et al., 2012). The bigger the size of data to be flashed, the longer it will take to compute its hash value (needed for signature verification). Consequently, deploying the framework on thousands of devices delays component delivery for days or even longer and minimizing the overhead of security mechanisms on the production process is beneficial. Another example that may benefit from the optimizations presented here stems from the fact that the platforms employed here are commonly used in the automotive industry. In-vehicle communication has recently become an active research area within the security community (Lemke et al., 2006). At the very least, secure communication between embedded devices relies on secure gateways (Wolf et al., 2006) that share secret keys and ultimately rely on MAC codes, i.e., keyed hashes. Obviously, many other examples for the use of hash functions can be envisioned.

Implementing cryptography on resource constrained devices is a well investigated subject and several solutions were successfully employed in practice. One category focuses on devising secure protocols which require little computational power and reduced variants of cryptographic functions. A good example in this area comes as a result of the intense research activity in sensor networks which produced solutions ranging from efficient protocol design to efficient cryptographic primitives (Karlof et al., 2004). Small scale variants of hash functions were also proposed for use in RFID environments which can be even more constrained than sensor networks (Macchetti et al., 2005). However, collisions on these functions were already reported (Steurer, 2006). Another category of solutions are based on hardware implementations. Using ASIC or FPGA-based cryptographic hardware to perform the computation of required primitives increases performance along with the costs of production. Dedicated cryptographic coprocessors were developed to accelerate the execution of different primitives. Examples of such hardware implementations can be found in (Okada et al., 2000) and (Suh et al., 2005). Some efforts were also made in enhancing the performance of general purpose microcontrollers by extending their instruction set with application-specific instructions used in cryptographic algorithms (Groschdl et al., 2004). Although they reach good performances, these hardware-based solutions are application dependent and require extra time to be spent on designing them in comparison to a software-based solution. Therefore, software solutions based on microcontrollers that are already available on the market may be preferred in various contexts.

Microcontrollers are constantly evolving to handle the need for increased performance. Different manufacturers are already offering microcontrollers with on-chip general purpose coprocessors or even dual core microcontrollers (Renesas Electronics, 2010; Freescale, 2009). This category of microcontrollers could be used to enhance cryptographic performance by using parallelism. One way to put this into practice is by running multiple instances of a function in parallel on each core to achieve high throughput. However, not all applications can rely on this kind of parallelism. Frequently, a single execution of a cryptographic primitive is needed at a certain time; so in order to attain speedups we have to search inside each individual algorithm for steps that can be parallelized. Some frequently used cryptographic algorithms were studied in this respect for the implementation of an FPGA-based crypto processor (Buchty et al., 2004) and similar solutions are needed for multicore microcontrollers. Coming to the

Location-Awareness with Action Systems

Luigia Petre, Kaisa Sere and Marina Waldén (2012). *Handbook of Research on Mobile Software Engineering: Design, Implementation, and Emergent Applications (pp. 463-483).*

www.igi-global.com/chapter/location-awareness-action-systems/66483?camid=4v1a

Putting Personal Smart Spaces into Context

Ioanna Roussaki, Nikos Kalatzis, Nicolas Liampotis, Pavlos Kosmides, Miltiades Anagnostou and Efstathios Sykas (2015). *Handbook of Research on Innovations in Systems and Software Engineering (pp. 710-730).*

www.igi-global.com/chapter/putting-personal-smart-spaces-into-context/117946?camid=4v1a