Architectures of Flexible Symmetric Key Crypto Engines—A Survey: From Hardware Coprocessor to Multi-Crypto-Processor System on Chip

LILIAN BOSSUET, University of Lyon
MICHAEL GRAND, University of Bordeaux
LUBOS GASPAR and VIKTOR FISCHER, University of Lyon
GUY GOGNIAT, University of South Brittany

Throughput, flexibility, and security form the design trilogy of reconfigurable crypto engines; they must be carefully considered without reducing the major role of classical design constraints, such as surface, power consumption, dependability, and cost. Applications such as network security, Virtual Private Networks (VPN), Digital Rights Management (DRM), and pay per view have drawn attention to these three constraints. For more than ten years, many studies in the field of cryptographic engineering have focused on the design of optimized high-throughput hardware cryptographic cores (e.g., symmetric and asymmetric key block ciphers, stream ciphers, and hash functions). The flexibility of cryptographic systems plays a very important role in their practical application. Reconfigurable hardware systems can evolve with algorithms, face up to new types of attacks, and guarantee interoperability between countries and institutions. The flexibility of reconfigurable crypto processors and crypto coprocessors has reached new levels with the emergence of dynamically reconfigurable hardware architectures and tools. Last but not least, the security of systems that handle confidential information needs to be thoroughly evaluated at the design stage in order to meet security objectives that depend on the importance of the information to be protected and on the cost of protection. Usually, designers tackle security problems at the same time as other design constraints and in many cases target only one security objective, for example, a side-channel attack countermeasures, fault tolerance capability, or the monitoring of the device environment. Only a few authors have addressed all three design constraints at the same time. In particular, key management security (e.g., secure key generation and transmission, the use of a hierarchical key structure composed of session keys and master keys) has frequently been neglected to the benefit of performance and/or flexibility. Nevertheless, a few authors propose original processor architectures based on multi-crypto-processor structures and reconfigurable cryptographic arrays. In this article, we review published works on symmetric key crypto engines and present current trends and design challenges.

Categories and Subject Descriptors: C.3 [**Special-Purpose and Application-Based Systems**]: Microprocessor Applications; C.1 [**Processor Architectures**]

General Terms: Security

Additional Key Words and Phrases: Cryptosystems, reconfigurable architecture, crypto processor, crypto coprocessor, crypto array, crypto MPSoC

This work is supported both by the French National Recherche Agency (ANR), project SecReSoC ANR-09-SEGI_013 and by the French General Armaments Directorate (DGA). The views expressed in this article are those of the authors and cannot be regarded as stating and official position of the DGA or the French DoD.

Authors' addresses: L. Bossuet (corresponding author), Hubert Curien Laboratory, UMR 5516 CNRS, University of Lyon at Saint-Etienne, France; email: lilian.bossuet@univ-st-etienne.fr; M. Grand, IMS Laboratory, UMR 5218 CNRS, University of Bordeaux; France, L. Gaspard and V. Fischer, Hubert Curien Laboratory, UMR 5516 CNRS, University of Lyon at Saint-Etienne, France; G. Gogniat, Lab-STICC Laboratory, UMR 3192 CNRS, University of South Britanny at Lorient, France.

DOI: http://dx.doi.org/10.1145/2501654.2501655

ACM Reference Format:
Bossuet, L., Grand, M., Gaspar, L., Fischer, V., and Gogniat, G. 2013. Architectures of flexible symmetric key crypto engines—A survey: From hardware coprocessor to multi-crypto-processor system on chip. ACM Comput. Surv. 45, 4, Article 41 (August 2013), 32 pages.
DOI: http://dx.doi.org/10.1145/2501654.2501655

1. INTRODUCTION

In recent decades, the issue of data security has shifted from the military to the commercial arena (e.g., banking, communications, networking, and multimedia systems) [Anderson et al. 2006]. The need for cryptographic computation has increased exponentially and the need to build powerful cryptographic computing resources has never been greater. As a consequence, many research teams have designed efficient hardware crypto engines that implement cryptographic primitives, algorithms, and/or protocols.

In this article, we present some aspects of published works on flexible symmetric key crypto engines. By flexible crypto engines we mean different kinds of systems such as customized general-purpose processors, hardware crypto coprocessors (or hardware crypto accelerators), crypto processors, and crypto arrays (dedicated coarse-grained reconfigurable architectures composed of an array of small cryptographic processing elements). Some of the works presented here are not flexible in the strictest sense of the term, but they could be, if they were implemented on a reconfigurable platform such as a Field Programmable Gate Array (FPGA). Designers of crypto engines try to meet the three basic requirements of a security application: throughput-flexibility-security, while other design constraints such as area, power consumption, dependability, and cost must also be taken into account, but in some cases to a lesser extent. Historically speaking, applications such as VPN or network security focused on the first three basic constraints. Here we analyze existing hardware architectures and their design space in the context of these three constraints.

The article, is organized as follows: in Section 1, we present the design space of cryptographic computing resources and their exploration. In Section 2, we briefly explore the crypto processor and the coprocessor jungle. In Section 3, we provide an overview of all possible solutions. In Section 4, we give an alternative solution in the form of a multi-crypto core processor and in Section 5, we highlight current crypto engine design challenges. In Section 6, we present a number of conclusions.

2. THROUGHPUT/FLEXIBILITY/SECURITY TRADEOFF

Nowadays, many options are available for designing crypto resources. All of them describe the design space of the cryptographic resources. This design space is finite but not static. Indeed, advanced technologies can extend it over time.

Not very long ago, designers had two obvious but opposite choices: using a General Purpose Processor (GPP) or designing an Application Specific Integrated Circuit (ASIC). Using a GPP is a very flexible solution, but is generally not appropriate for high-performance applications, because of the sequential execution of the algorithm and the limited fixed data path width.

The speed of the system can be increased if the GPP is replaced by a crypto processor. Its instruction set and Arithmetic Logic Unit (ALU) can be optimized for the execution of cryptographic algorithms. If necessary, throughput can be further enhanced at the expense of flexibility using one or more hardware accelerators (or coprocessors). Many coprocessors have been designed to perform crypto-dedicated computations/algorithms. Like in digital signal processing, dedicated accelerators significantly increase the speed of data processing in cryptographic applications, but their use is limited to the algorithm/architecture used.

To further enhance performance, designers can fully implement their application in hardware using ASIC technology. This can achieve the best performance, but may be expensive for small runs and also requires careful management of hardware resources (choice of the width of the data path, registers, and memory size, etc.), because its hardwired structure means it cannot be updated.

Designers of cryptographic applications are under constant pressure due to security requirements, such as data confidentiality, integrity, and authenticity. Since this article, deals with hardware architectures for symmetric key cryptography, data security is considered as the ability of a system to protect information and system resources, especially data confidentiality. Since security is never cost free, designers have to adjust the targeted security level to security objectives, which depends on the profile of the attacker, the physical accessibility of the device to the attacker (remote or physical attacks), and the value of the information to be protected [Badrignan et al. 2011]. During the design, security threats must be considered at all levels of abstraction: application and protocol, software, macro-architecture, logic, and physical levels.

Anderson [2001] reported that a protocol weakness enabled an attack on an IBM 4758 crypto processor previously thought to be secure. Along with software attacks and cache memory attacks, protocol attacks are very dangerous, because the attacker does not need physical access to the device. Bangerter et al. [2011] showed that small malicious software can monitor the cache memory during enciphering and the key can be recovered remotely in a few minutes. Protocol and software attacks can be countered at architecture level by preventing processors having direct access to security-critical parameters such as keys [Gaspar et al. 2010, 2011]. Logic-level countermeasures such as data hiding and data masking are aimed at protecting the devices against side-channel attack techniques [Standaert et al. 2003]. Side-channel attacks are very powerful and consequently very attractive, but to undertake them, the attacker needs physical access to the device. In addition to logic-level countermeasures like dual rail logic, some technological countermeasures like those proposed in Tiri and Verbauwhede [2005] can also reduce information leakage.

The designer can construct a circuit that includes countermeasures against known physical attacks (e.g., side-channel and fault injection attacks). But the countermeasures implemented may subsequently be jeopardized by new attacks and the system will consequently need updating. Unfortunately, updates are impossible using ASICs. In some cases, countermeasures, which should increase the security of the device, can even be used as a source of leakage and facilitate the attack [Regazzoni et al. 2008]. When dealing with ASICs, security is clearly a major challenge and should be very carefully addressed in order to anticipate all possible attacks.

Since the early 2000s, data security algorithms have been more and more frequently implemented in FPGAs, because the granularity of configurable logic devices such as FPGAs and their architecture featuring small logic elements (able to implement 4- to 8-input combinatorial logic functions) suits the calculations used in many cryptographic algorithms [Wollinger et al. 2004; Wollinger and Paar 2003]. FPGA technology has considerably evolved in the last 12 years. In 2000, high-end FPGAs such as 0.22μm Xilinx Virtex embedded a 27k-logiccell (with 4-input LUTs) and 16KB of SRAM, which ran at 200MHz [Xilinx Corp. 2001]. Today, a typical high-performance 28nm FPGA (e.g., Xilinx Virtex-7) embeds more than 2M logic cell (with 6-input LUT), 50MB of SRAM, and 2k-DSP which runs at 700MHz, with many high-speed (12.5Gb/s) serial transceivers [Xilinx Corp. 2012]. Such capabilities make FPGAs significant targets for data security applications.

Furthermore, when using SRAM- or FLASH-based FPGA technology (around 90% of the market), designers can take advantage of their reconfiguration capabilities and let the system evolve over time by means of hardware updates made in situ and even

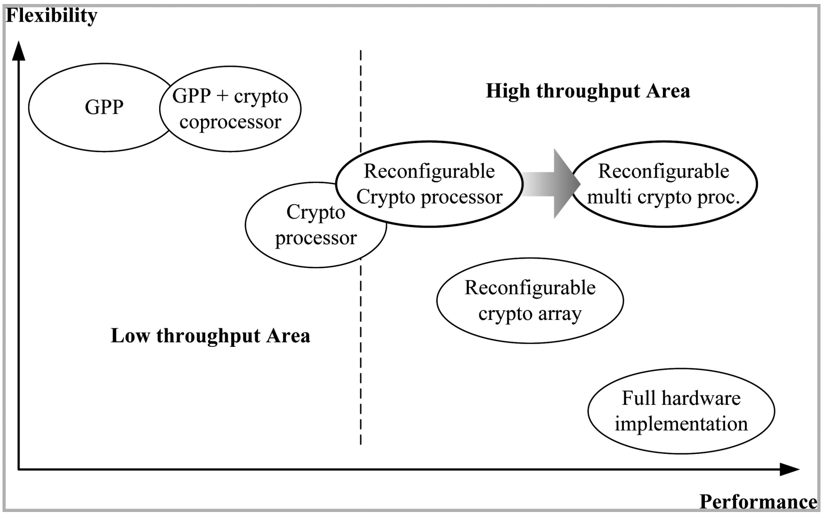

Flexibility
GPP
GPP + crypto coprocessor
High throughput Area
Reconfigurable Crypto processor
Reconfigurable multi crypto proc.
Crypto processor
Reconfigurable crypto array
Low throughput Area
Full hardware implementation
Performance

Fig. 1. 2D (performance/flexibility) cryptographic computing design space.

remotely [Davies 2003]: an obsolete algorithm can be replaced by the latest version, a new architecture embedding countermeasures against attacks can replace an old one if it is vulnerable to some types of attacks, and so on.

While the use of GPP, dedicated crypto processors, crypto coprocessors, and hardwired cryptographic blocks in FPGAs is still possible, reconfigurable technologies provided many new features for the design of cryptographic hardware. Softcore GPPs usually have a flexible instruction set and modifiable ALU architecture. The design of crypto processors can thus be considerably simplified and ensure increased flexibility, meaning that both hardware and software can be modified. A hardware accelerator implemented in FPGA is no longer considered as a rigid component, since it can be reconfigured whenever necessary (its hardware architecture can be modified). The designer can take advantage of the flexibility of this solution by using an appropriate hardware-software codesign tool when defining the boundary between hardwired and soft functions. Very recent work on high-performance (high-throughput) systems demonstrated the interest of continuing development in this area, while at the same time extending it towards multi-crypto-processor architectures [Grand et al. 2011]. The way is open to new concepts such as crypto array and MCryptoPSoC *(Multi-Crypto-Processor System-on-a-Chip)*.

Figure 1 is a theoretical view of a 2D cryptographic computing design space. The first dimension is “performance”. This general term describes the computation performance and the throughput (given, for example, in gigabits per second of encrypted data). The second dimension is “flexibility”. This is the ability of a cryptographic module to move from one algorithm/computation to another and takes the application dependency of the architecture into account.

GPPs and GPPs with a crypto coprocessor are the most flexible solutions because they are application independent. With such systems, it is easy to switch between applications merely by modifying the program memory content. On the other hand, full hardware implementations are more time efficient because they allow parallel, pipelined, unrolled, and optimized algorithm implementation. The relative position of each design solution in the figure shows that high flexibility and high throughput are contradictory requirements. Heterogeneous architectures such as crypto processors and reconfigurable crypto processors are located in the center of Figure 1. Nevertheless, new reconfigurable architectures based on multi-crypto processors can reach a new trade-off area to ensure a higher level of flexibility along with higher throughput.

One point is missing in Figure 1: the third dimension, security. Indeed, studies that compare the security levels of available architectures are rare in the literature Badrignan et al. [2011]. Yet security is increasingly being taken into account. For example, in Section 2.3 we provide some details about a secure crypto-processor architecture, *HCrypt*, to better protect session keys against illegal access.

In the following section, we illustrate the design space of cryptographic resources using some concrete examples.

3. FINDING THE WAY THROUGH THE CRYPTO-PROCESSOR AND COPROCESSOR JUNGLE

As the use of the word "jungle" in the heading suggests, it is difficult to get a clear overview because of the profusion of publications on cryptographic computing. This is even truer since the terminology is constantly changing and the terms "custom processor", "crypto processor", "crypto coprocessor", and "crypto array" were never clearly defined. Figure 2 shows four possible ways of overcoming this problem: customized GPP, crypto coprocessors, crypto processors, and crypto arrays. The following sections give the characteristics of each option and refer to works that illustrate the architectures concerned. The reports cited were published between 1999 and 2011.

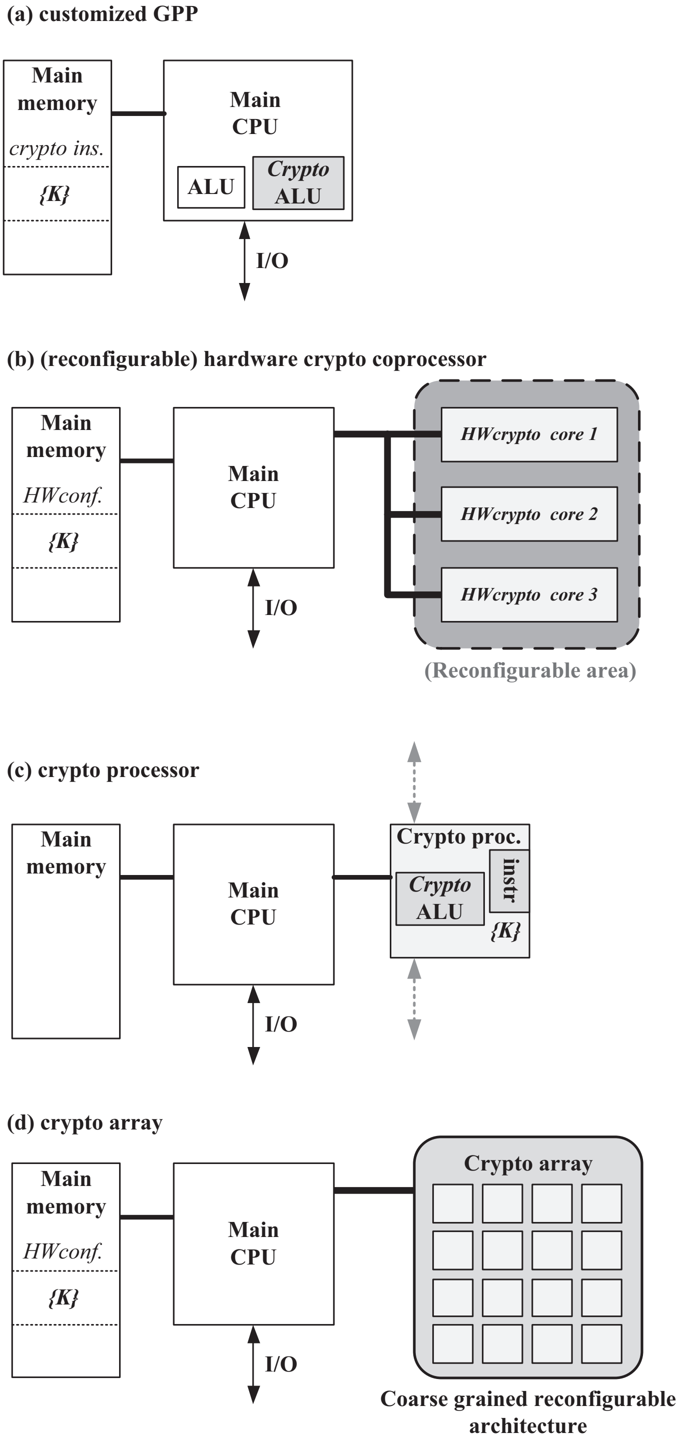
At the end of each of the following sections describing different types of crypto engines, we discuss their future outlook in terms of their use, advantages, and disadvantages as objectively as possible.

3.1. Customized GPP

Figure 2(a) is a schematic diagram of a general-purpose processor that is customized for efficient implementation of cryptographic algorithms. It embeds a functional unit able to perform a number of cryptography-specific operations, such as Data Encryption Standard (DES) logical computations or Advanced Encryption Standard (AES) substitution functions *(S-Box)*. In this case, the cryptographic calculations are seen as application-specific instructions that can be called on during program execution. This solution increases throughput, but not security. Confidential keys are stored in the data memory and are handled just like ordinary application data. Software attacks cannot be avoided.

One of the main difficulties when designing a domain-specific processor is selecting an appropriate specific instruction set. Asymmetric key algorithms and symmetric key algorithms do not have the same requirements. Asymmetric key ciphers use multi-precision operations whereas symmetric key ciphers use bit-level operations. Moreover, the specific instruction area overhead is sometimes too large for realistic implementation. To design a customized processor for efficient crypto-specific processing, Ravi et al. [2002] proposed a usual codesign flow applied to a security processor. The authors used two models: a performance macromodel based on functions that express the number of cycles incurred by a software library, and a hardware model. They proved that a customized processor designed using the method they proposed is significantly more efficient than full software implementation on a GPP. This was demonstrated on a customized 32-bit Xtensa processor from Tensilica. Since the first proposal to use HW-SW codesign methodology on a customized processor aimed at cryptographic applications, other authors have proposed similar approaches but they target superscalar architectures [Sakiyama et al. 2007] or embedded systems [Schaumont and Verbauwhede 2003].

Several authors extended the processor's instruction set to enable efficient AES computation on a GPP [Hämäläinen et al. 2007]. The first works focused on S-Box implementation using dedicated instructions [Burke et al. 2000; Tillich et al. 2005] and



(a) customized GPP

(b) (reconfigurable) hardware crypto coprocessor

(c) crypto processor

(d) crypto array

Fig. 2. Cryptographic computing architecture design space.

enhancing the performance of S-Box computation. Next, Tillich et al. designed a GPP-based system using several custom instructions, thereby enhancing the performance of the whole AES implementation [Tillich and Grobschad 2006]. These instructions were developed in such a way that they were compatible with common processors based on 32-bit RISC architecture (e.g., SPARC, MIPS, and ARM processors). Instruction complexity ranged from simple AES *S-Box* [Tillich and Grobschad 2005] and AES *MixColumns* implementation to composite AES *S-Box*/*MixColumn*s/*ShiftRow*s execution [Tillich and Grobschad 2006; Tillich and Herbst 2008]. Even though this approach improved the performance of a GPP-based system, it was mostly dedicated to only one algorithm (e.g., AES).

Another design, the *CryptoBlaze* processor from Xilinx [Xilinx Corp. 2003] is an interesting example of what is available commercially. It is based on the use of a *PicoBlaze* softcore processor, which is a compact, cost-effective 8-bit RISC microcontroller core optimized for Xilinx FPGA families [Xilinx Corp. 2010]. The *CryptoBlaze* processor system represents a case study of expanding the *PicoBlaze* processor with additional functions/opcodes aimed at cryptographic applications. In the first version, the Galois field arithmetic and *S-Boxes* were implemented as specific instructions. The *CryptoBlaze* architecture is an interesting combination of a reconfigurable fabric and a customized microcontroller for cryptographic applications. Another commercial example was presented recently by Intel [Gueron 2010]. The instruction set of some recent company processors based on the 32nm Intel micro-architecture is extended by six instructions dedicated to AES implementation. These instructions drive a dedicated AES ALU aimed at data enciphering, deciphering, and key expansion. To perform AES modes, such as ECB, CBC, or CTR, some libraries that use this specific set of instructions are available. This shows that cryptography has become a key element of computer architecture, as was the case of image processing in video processors several years ago.

Discussion: ALUs dedicated to cryptographic calculations are especially suitable for applications with stringent area constraints (limited area budget requirements), and in particular in embedded applications. Dedicated ALUs allow the designer to benefit from processor flexibility and at the same time to accelerate cryptographic computations. The hardware overhead is low and the design can be relatively simple when the host processor design is based on an open-source soft-core processor. However, in all previously presented configurations, the ALU architecture is application specific and its reusability is very limited, because its structure is closely linked to the target cryptographic tasks and also because its data interface depends on the host processor. In addition, new instructions sometimes have to be added to the processor instruction set. Extending the instruction set remains a major problem for optimized use of hardware resources since the compiler has to be rebuilt to take the new instructions into account. Calculations are faster since data are transferred via the processor's internal data path (they do not need to be sent to the external memory or peripheral bus), but the processor architecture remains sequential, and calculations cannot be further accelerated by parallelizing tasks. This reduces the use of customized processors for high-throughput applications. It should also be noted that these solutions are not compatible with a fine runtime reconfiguration of the cryptographic ALU because the corresponding logic is in the processor's internal data path.

3.2. Crypto Coprocessors

Figure 2(b) shows a multi-algorithm custom hardware implementation. Such a hardware implementation always provides the highest level of efficiency. However, in practice, the need for efficiency in data security applications often needs to be considered together with and as an integral part of a trade-off against the need for flexibility.

With this strategy in mind, it is possible to implement flexible hardware accelerators when using (dynamic) reconfiguration capabilities of selected FPGAs. In this way, both efficiency and flexibility can be obtained. However, none of the coprocessors presented here is reconfigurable. Like in customized GPP systems, secret keys and hardware configuration are stored in the data memory.

The terms crypto processor and crypto coprocessor are occasionally reported in the literature to be used in different contexts, which sometimes causes confusion. To avoid any such misunderstandings, here we distinguish the two processor systems according to their programmability. In this sense, the crypto processor is a programmable device or programmable hardware module, with an instruction set dedicated to the efficient implementation of cryptographic function(s). It thus mostly consists of one or more ALUs specially designed for cryptographic computations. On the other hand, the crypto coprocessor (coprocessor for short) is a logic device or hardware module dedicated to the execution of cryptographic function(s). It contains one or more processing elements. The crypto coprocessor cannot be programmed, but can be controlled, configured, or parameterized using the host processor. It is used to accelerate cryptographic computations. To illustrate this point, two early publications targeting the DES algorithm provide typical examples of crypto processors and coprocessors: a DES crypto processor, which is programmed using an instruction set, was proposed by Verbauwhede et al. [1991] whereas a DES coprocessor, which is a scalable hardware implementation on an FPGA, was proposed by Kaps and Paar [1998].

An example of a single-core AES coprocessor is described in Hodjat and Verbauwhede [2004a, 2004b, 2005]. These authors examined the addition of the AES coprocessor to a SPARC V8 embedded processor (LEON), and showed that the benefits of a hardware accelerator can be significantly reduced by a communication overhead (i.e., the transfer of data to and from the coprocessor).

Two examples of multicore AES coprocessors driven by a Molen processor [Kuzmanov et al. 2004] are described in Chaves et al. [2006] and Pericàs et al. [2008]. The first structure, proposed by Chaves et al., is composed of a GPP and several reconfigurable CrCU (Cryptographic Computation Units) interconnected by a dedicated network [Chaves et al. 2006]. Each CrCU embeds a full AES core, a key register, and a control unit. Figure 3 shows the overall architecture of the multicore system. The AES CrCU and the Molen processor communicate with each other using a standard address and data bus. The AES CrCU is also connected with the main memory to exchange data (to/from the cipher) and cipher key (to the internal key register). According to the authors, the CrCU could be reconfigurable. The second structure proposed by Chaves et al. is a dual AES core hardware implementation for AES MultiStream (AES-MS) ciphers [Pericàs et al. 2008]. It consists of two independent AES cores controlled by a control unit, which activates the AES cores when required and drives the multiplexors that control access to the data bus.

An older work proposes a modular coprocessor called *CryptoBooster* dedicated to cryptography and optimized for reconfigurable logic devices such as FPGAs [Mosanya et al. 1999]. It is a reconfigurable coprocessor with a host system and is connected to a dedicated session memory designed to store session information (e.g., the selected block cipher algorithm, the key(s), the initial vector(s) for block chaining, and other configuration information).

Although most studies in this field focus on performance, some original works were dedicated to optimizing the entire system and its security. For example, the *SAFES* (Security Architecture For Embedded Systems) architecture by Gogniat et al. [2008] depicted in Figure 4 resembles the multicore coprocessor architecture in Figure 2(b). In addition to its similar structure, this system uses monitors that detect abnormal behavior. Hardware defence mechanisms can be implemented to counter attacks. The

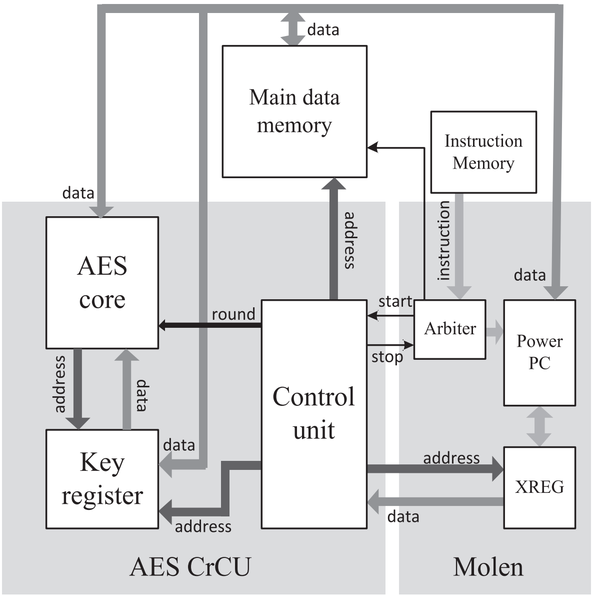


Fig. 3. Architecture of a crypto system with the Molen processor and dedicated hardware accelerator AES CrCU [Chaves et al. 2006].

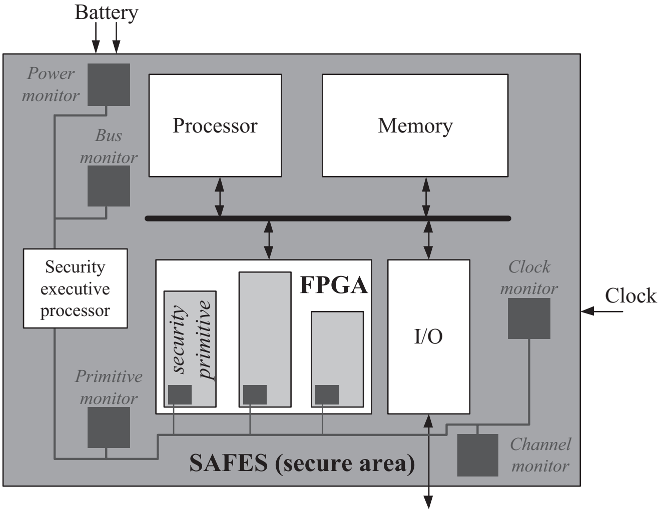


Fig. 4. *SAFES* reconfigurable secure architecture [Gogniat et al. 2008].

security mechanisms can be updated (dynamically) if necessary, which ensures the durability of the protection system.

As can be seen, several monitors can be used to monitor processes that indicate or trigger an attack on the system. Parameters such as the number of monitors and the complexity of individual monitors are important, because they directly affect the additional cost of security features as well as the level of security provided. For security reasons, the normal activity of the modules under supervision is characterized and continuously compared with the current activity of the system. The monitors are autonomous so they can contribute to the fault tolerance of the system; if one monitor is attacked, the others must be able to continue to function in order to guarantee the security of the system. The monitors are distributed at different locations in the system to detect security-critical events (e.g., those concerning the battery, bus, security primitives, and communication channels).

The *SAFES* system can react at different levels depending on the type of attack underway. Instant reactions are triggered directly by a corresponding monitor without consulting the other security units. Global reactions are executed when an attack involves a significant modification to the system. In this case, the monitors exchange information in order to define a new configuration. Such a scenario allows more complex attacks to be detected, but makes the reaction time longer. The reconfigurable part (FPGA) within the system allows hardware implementation of security primitives. This leads to an adaptive hardware accelerator operating a security-related algorithm (cipher, hashes, key management). In contrast with the aforementioned crypto coprocessors, the list of supported algorithms is not fixed. The user configures the system with the security primitives required. These can be updated by reconfiguring them during the lifetime of the system. Thus, the *SAFES* architecture provides the performance (hardware implementation) and flexibility (reconfigurable system) needed to secure embedded systems.

Another example of a multicore AES coprocessor close to the architecture in Figure 2(b), is the *AESTHETIC* system [Wang et al. 2010]. Based on the single *AESTHETIC* coprocessor [Su et al. 2005] the authors propose a high-performance multicore architecture, in which independent data paths for each *AESTHETIC* coprocessor allow multigigabit security processing with no loss of I/O bandwidth.

Remark # 1. Some interesting works have considered the use of a GPU as a crypto processor combined with a CPU [Cook et al. 2005; Manavski 2007; Deguang et al. 2010]. Such approaches are motivated by the widespread adoption of GPUs in many embedded systems. Using a GPU for symmetric cryptography enables high throughput of up to 8.28 Gbit/s [Manavski 2007]. The performance is directly related to the data size (i.e., the number of blocks) to be encrypted and exploits the parallelism available in GPUs. In all published works, the key expansion step is performed in the CPU and the keys are then loaded into the GPU before encryption. Such a solution has certain limitations concerning the ciphering modes as only ECB and CTR can benefit from computing blocks in parallel. The exchange of keys between CPU and GPU may be a concern for some applications and should be analyzed very carefully. Power issues were not addressed in Cook et al. [2005], Manavski [2007], and Deguang et al. [2010], but this point should also be analyzed to evaluate its impact on the global power budget.

Remark # 2. Several secure processors with embedded hardware cryptographic primitives are described in the literature. The need to protect data for reasons of confidentiality, integrity, and authenticity is obvious. However, the program code also needs to be protected for the same reasons to limit possible software attacks (e.g., spoofing attack, splicing attack, replay attack). At present, proposed solutions rely on an inviolable hardware security zone called a *trust zone, trust area,* or *secure area*, which usually contains the processor, cache memory, data and program memories, and the memory access controller. This zone includes the hardware protection systems to protect the trust zone against known hardware attacks at physical, logical, and architectural levels. Data leaving and entering the trust zone are encrypted and authenticated to ensure protection at system level. The trust zone contains hardware primitives to enable ciphering and authentication of the data (e.g., instructions, addresses, and data) exchanged between the processor and its memories. Unlike the architecture depicted in Figure 2(b), in the case of a trust zone, the secret keys are stored inside the processor and not in the external memory. In most cases, a Physical Unclonable Function *(PUF)* or a True Random Number Generator *(TRNG)* is used to generate a master key that is often used with a hierarchical key tree. *XOM* [Lie et al. 2000], *AEGIS* [Suh et al. 2003], *TSM* [Lee et al. 2005], *PE-ICE* [Elbaz et al. 2006], *CryptoPage* [Duc and Keyrell 2006, 2008], and *OTP-CRC* [Vaslin et al. 2007] are representative examples of such an approach. It is important to note that, even if the codes and the data are protected in the external memory, in some cases these solutions can still be bypassed. Indeed attacks exploiting micro-architectural features like distribution of cache hits and misses can be very powerful and must be carefully analyzed [Bernstein 2005; Osvik et al. 2006; Rebeiro et al. 2009; Rebeiro and Mukhopadhyay 2011]. Another threat is related to signature addresses when a specific application is executed. In Zhuang et al. [2004], bus protection is used to remove all data leakage between the processor and the external memory when spying on the address bus. The approach suggested by Zhuang et al. [2004] is based on analysis of the control-flow graph, which provides a unique signature for the application. Thus in order to hide such information, code address transformation is performed using permutation techniques. One last point that needs to be stressed when dealing with external memory is data persistence, which must also be considered with great care [Halderman et al. 2009].

Discussion. The hardware crypto coprocessor can speed up cryptographic computation by using the inherent parallelism of the algorithm. Many iterative cryptographic algorithms can be unrolled. For example, most symmetric ciphers use the same elementary logic/arithmetic computations in their subsequent rounds. The AES block cipher uses 10 to 14 rounds, each composed of four functions: *SubBytes*, *ShiftRows*, *MixColumns*, and *AddRoundKey*. Hardware implementation can thus take advantage of the inherent parallelism of the unrolled cipher. As a consequence, hardware crypto coprocessors are more suitable for expensive high-throughput applications (network security, VPN) than for small low-cost systems. Nevertheless, the interconnection between the processor and coprocessor can seriously limit the performance of the entire system. If the processor has a direct link to the coprocessor, it can access data very fast. Unfortunately, these links are often limited in size, because the width of the data interfaces between the processor and coprocessor have to match. The use of a communication bus is of course possible, but the time overhead can be prohibitive. Everything depends on the frequency of exchanges between the processor and the coprocessor. For example, the symmetric key cryptography cipher modes (e.g., CBC, EBC, CTR, CBC-MAC, etc.) can be implemented in the hardware coprocessor whenever possible. We may add that the implementation of a hardware coprocessor is consistent with the notion of reconfiguration. Indeed, the runtime reconfiguration of the coprocessor can take place as soon as the processor releases the coprocessor from computation. This can increase the flexibility of the whole system and limit the size of hardware resources by sharing many algorithms. In this case, special attention should be paid to configuration management.

3.3. Crypto Processor

Figure 2(c) shows a GPP coupled with a dedicated cryptographic processor or crypto processor. According to the definition at the beginning of the previous section, the crypto processor embeds one or many ALUs dedicated to cryptographic computations. As a consequence, the crypto processor cannot serve as a stand-alone general-purpose processor. It is always necessary to extend the system by adding a GPP to perform standard computations or to embed an operating system. Such architecture can be smaller than the GPP and crypto coprocessor and it may be more flexible if the ALUs are properly reconfigured. Crypto processors can be used for many applications, for example, for red/black radio systems [Martin et al. 2008] or as a crypto subsystem [Grand et al. 2009].

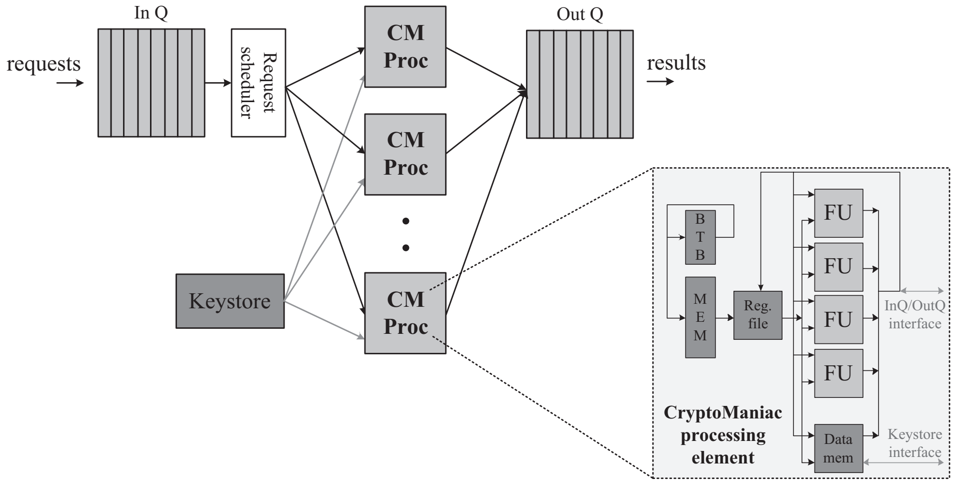


Fig. 5. The *CryptoManiac* system and its processing element architecture [Wu et al. 2001; Weaver et al. 2001].

Several papers describe designs for Very Long Instruction World (VLIW) crypto processors aimed at symmetric cipher implementation. Such algorithms can use several key sizes (e.g., 128, 192, 256 bits), but usually use smaller data blocks for computation (e.g., 8, 16, 32 bits). Based on this characteristic, it is efficient to design superscalar architecture with multi-operand instructions.

The *CryptoManiac* project is an example of a VLIW crypto processor able to execute up to four 32-bit instructions per clock cycle [Wu et al. 2001; Weaver et al. 2001]. This multicore crypto processor uses many *CryptoManiac* processing elements (CM). Each is designed with four dedicated *Functional Units* (FU), one *Branch Target Buffer* (BTB) used to predict branch targets and register the file and memory. Figure 5 shows the *CryptoManiac* system architecture and its processing element architecture. Additionally, short-latency instructions can be combined to be executed in a single cycle. To support this feature, instructions have up to three source operands. The *CryptoManiac* system consists of four functional units that access a shared memory. Another four 32-bit instruction VLIW crypto processors are available in the *CCProc* Project [Theodoropoulos et al. 2008, 2009]. This recent work targets larger algorithmic space than *CryptoManiac*.

The *Cryptonite* project is another example of VLIW architecture with two 64-bit data paths [Buchty et al. 2004]. It supports AES round functions across a set of special instructions for performing *byte permutation*, *rotation,* and *xor* operations. The main part of the AES algorithm is executed with the help of parallel LUTs (Look-Up Tables) stored in dedicated memories. The *Cryptonite* architecture has two ALUs and separate memory units for each ALU. As a consequence, the address generation unit and the data unit are duplicated.

Many common security devices are based on a general-purpose process communication protocol and on a dedicated hardware coprocessor to speed up cryptographic algorithms. However, security is immediately compromised if secret keys are stored in data registers or handled by a processor [Bangerter et al. 2011]. According to Gaspar et al. [2010], a secure cryptographic processor should fulfill the following requirements.

(1) Keys have to be generated inside the crypto processor and postprocessed cryptographically.

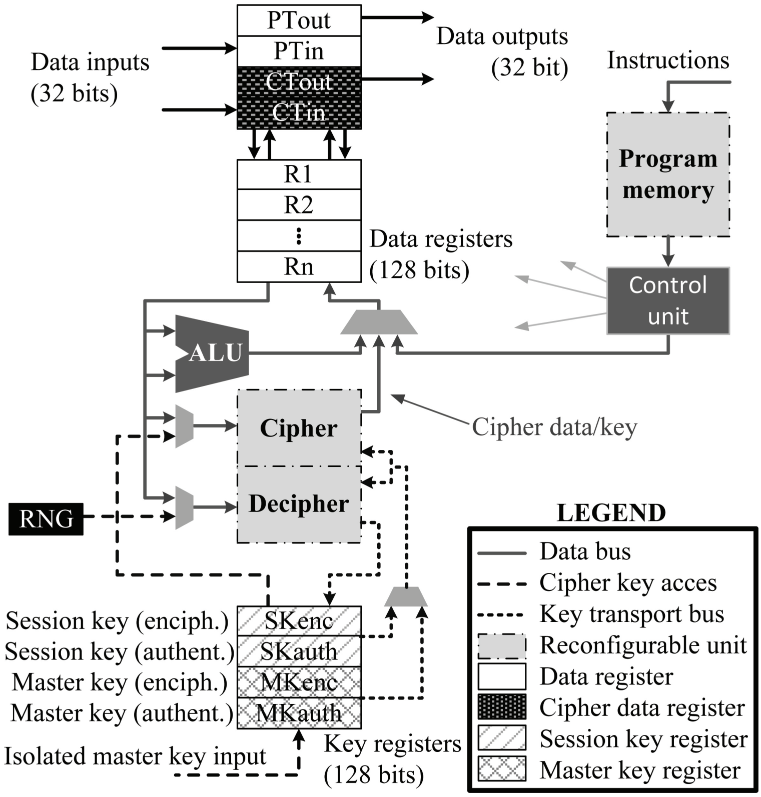


Fig. 6. Architecture of *HCrypt* crypto processor [Gaspar et al. 2010].

(2) Keys have to be stored in a dedicated memory which is physically separated from the data memory.
(3) Keys must be transported within the processor using dedicated (key) buses, which have to be physically separated from the data buses.
(4) The processor can read from/write to the key memory via the cipher/decipher, so that the keys can never leave system in clear.

The structure of the *HCrypt* crypto processor that fulfills these requirements is illustrated in Figure 6. The proposed solution is secure against software attacks thanks to its unique architecture. The data-path architecture of the processor can be divided into two zones: a protected zone (key transport and storage) and an unprotected zone (data storage and processing). These zones are physically separate from each other. This complete separation of two security zones ensures that keys stored in key registers (protected zone) can never pass unencrypted to data registers (unprotected zone) and hence leave the system in clear. The processor supports symmetric cryptography key exchange mechanisms based on two-level key hierarchy: session keys and master keys. Session keys are generated by the TRNG (located within the protected zone inside the processor); they are postprocessed by the decipher unit (using a master key) and safely stored in session key registers. During key exchange, a session key is encrypted with a master key and subsequently stored in a data register. In its encrypted form, the session key can be safely exchanged with other communication counterparts. The only condition that has to be met is that all the counterparts have to own the same master key. It is important to underline the fact that no such physical path exists through which session keys could be passed to data registers in unencrypted form; session keys have to pass through the cipher. Master keys are introduced into the processor by an isolated external bus. The processor is capable of carrying out basic cipher modes (EBC, CFB, OFB, and CTR) and communication is packet oriented. All the operations required for implementation of cryptographic modes are performed by an ALU, except for enciphering and deciphering, which are performed by the cipher and decipher units. All the components and processor itself have a 128-bit data path.

The level of key management security introduced by this concept exceeds the requirements of common consumer electronic applications and can also be used in aviation or military applications. This novel concept of key management is very promising and could be the beginning of a new era in the design of cryptographic processors.

Remark. The aim of many commercial crypto processors including *NEC MP211* [Arora et al. 2006] is the construction of mobile embedded systems or *TPMs* (Trusted Platform Modules) for trusted computing [TCPA 2003]. These systems are based on architectures that are usually simpler than those described before [Anderson et al. 2006]. However, a true reconfigurable crypto processor designed for these applications is not yet available. Some studies show that it could be advantageous to merge the concept of the TPM with that of FPGAs [Glas et al. 2008; Eisenbarth et al. 2007a, 2007b]. However, to our knowledge, no commercial version exists at present.

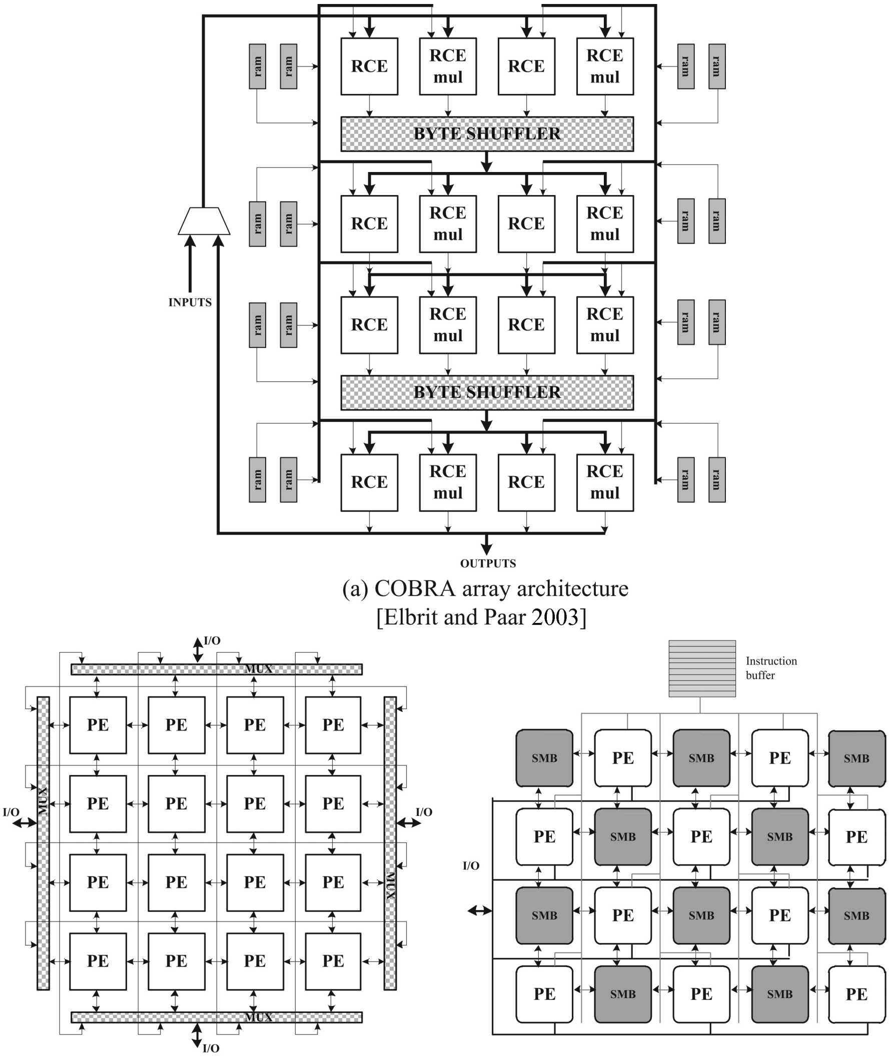
Discussion. Crypto processors are well-suited to be included in a Multi-Processor-System-on-a-Chip (MP-SoC). In SoC, they are considered as dedicated processors, just as DSPs are. They are autonomous even if they cannot execute anything other than cryptographic computations. They represent efficient solutions in terms of a trade-off between flexibility and performance even though their performance is limited by their size and architecture, which is optimized for the sequential execution of operations. Multicore systems are pushing the limits of performance and provide solutions for applications that need heterogeneous cryptographic computations. It is possible to reconfigure them if they are implemented on reconfigurable hardware (i.e., FPGA) [Gaspar et al. 2011].

3.4. Crypto Array

Figure 2(d) shows a GPP coupled with a cryptographic fabric. This is a coarse-grained reconfigurable architecture designed for cryptographic computing. Like crypto processors, a crypto array is a crypto accelerator that has to be coupled to a GPP.

Since the early 2000s, reconfigurable architectures have facilitated the appearance of new paradigms in computer architectures. Many authors have shown that such architectures are efficient for intensive computation applications [Tredennick and Shimamoto 2003], for embedded systems [Garcia et al. 2006], and for data security [Mucci et al. 2007]. In the mid-2000s, it was shown that an application-specific reconfigurable architecture is more efficient than a universal reconfigurable architecture for classical digital signal processing and for symmetric cipher use [Bossuet et al. 2005]. The following three examples illustrate this type of approach to implementing dedicated reconfigurable architecture for the AES cipher.

The first example, the *Celator* architecture (see Figure 7(b)), is based on a systolic array composed of 4×4 8-bit PEs (Processing Elements). The architecture processes operate on the AES *state* matrix (128-bit data) [Fronte et al. 2008]. Its structure is optimized for fast parallel computation of the AES algorithm. Nevertheless, it can perform other symmetric cipher algorithms such as DES and hash algorithms such as SHA-256. The routing of the *Celator* PE array is configurable and it can easily perform functions such as AES *ShiftRows*. Dedicated control logic can configure either the entire architecture or each PE individually. Performance comparisons with



(a) COBRA array architecture [Elbrit and Paar 2003]

(b) Celator array architecture [Fronte et al. 2008]

(c) CryptArray chessboard architecture [Lomonaco 2004]

Fig. 7. Examples of crypto array architectures.

GPP and customized GPP are in favor of *Celator*, however, its designers omitted the implementation of the key expansion system, which is very area and time consuming.

Similar to *Celator* but with larger processing (reconfigurable) elements, *COBRA* is a specialized reconfigurable architecture optimized for the implementation of block ciphers such as DES and AES [Elbirt and Paar 2003]. Its design is based on the analysis of the common functional requirements of a wide range of block ciphers (more than 40 algorithms were analyzed). It is designed with 4×4 32-bit RCEs (Reconfigurable Cryptographic Elements) (see Figure 7(a)). RCEs can execute the general operations required by the block ciphers (i.e., boolean, modular addition/subtraction, shift, rotation, modular multiplication, GF multiplication, modular inversion, and look-up table substitution). Moreover, a dedicated data path that can be configured as a substitution/permutation network is designed similar to the algorithm execution.

Another crypto fabric, called *CryptArray,* was proposed for a systolic architecture that uses shared memories to exchange data between PEs (Processing Elements) [Lomonaco 2004]. Four by four, the PEs are connected to one SMB (Shared Memory Block). If we consider each PE as a white tile and each SMB as a dark-gray tile, the *CryptArray* looks like a chessboard (see Figure 7(c)). Like *Celator*, *CryptArray* is a study of concept. Both crypto arrays are designed only for the enciphering process and do not account for other cipher parts such as the key scheduler.

Remark 1. Wollinger et al. showed the advantage of using FPGAs for cryptographic applications [Wollinger et al. 2004; Wollinger and Paar 2003]. For a decade, a huge number of papers described FPGA implementations of cryptographic algorithms. Countless conference sessions focused on such implementations. Nevertheless, all the previously presented crypto arrays are based on coarse-grained reconfigurable architectures; up to now, none considered fine-grained architectures. *Field-Programmable-Crypto-Array* or *Crypto-FPGAs* have not yet been developed. Elbirt and Paar [2003] gave an explanation for such a design trend: "because block ciphers are dataflow oriented, a reconfigurable element with coarse granularity offers the best solution for achieving maximum system performance when implementing operations that do not map well to more traditional fine-grained reconfigurable architectures". As was the case in the evolution of FPGAs towards signal processing applications because of the revolutionary growth of image processing (HDTV) and telecommunication markets, the evolution of FPGAs towards cryptographic applications is guided by the strong demand. Some trends are already apparent. Recently, Altera proposed a special version of its Cyclone III FPGA family (Cyclone III LS family [Altera 2011]) that is dedicated to military applications. The family includes several security features, such as antitampering, 256-bit AES bitstream enciphering, CRC, etc. The family and related design tools include features required for red/black security zone separation.

Remark 2. In the same way as it is necessary to secure the processor program code (see Remark 2 in Section 3.2), the configuration data (bitstream) of a reconfigurable device also needs to be protected for reasons of confidentiality, integrity, and authenticity in order to limit possible software attacks (e.g., spoofing attacks, splicing attacks, replay attacks). Some academic works have proposed adding (dynamically or statically) security features to protect the bitstream in a configurable device. Regarding SRAM and FLASH FPGA targets, the following references are of interest [Bossuet et al. 2007; Castillo et al. 2006; Güneysu et al. 2007; Hori et al. 2008; Nakanishi 2008; Manipatlolla and Huss 2011]. Most modern FPGAs embed bitstream security features. Nevertheless, the security of purchased FPGAs could be tampered with through existing attacks [Morabi et al. 2011, 2012].

Discussion. Figure 7 shows that the architectures of the three works mentioned before are very similar. Indeed, the three proposals are based on characteristics such as data-path width and the data dependence of symmetric block ciphers. For example, 128-bit AES performs computations on a byte level (for *SubBytes, ShiftRows*, and *AddRoundKey* functions) and on a 32-bit word level (for *MixColumns* function). As a consequence, a 4×4 byte matrix is currently used to directly implement AES computations. Reconfigurable hardware can be designed to compute data by using a 4×4 byte matrix array. In other words, a crypto array can be very close to the algorithmic and functional needs of the cipher. Nevertheless, a symmetric cipher is composed of two parts, the cipher itself and the key expansion. This last point is not taken into account in the three architectures depicted in Figure 7. Moreover, the use of such architecture requires a dedicated tool for the design of a coarse-grained architecture and a specific method to manage the reconfiguration. The main difference between the three works

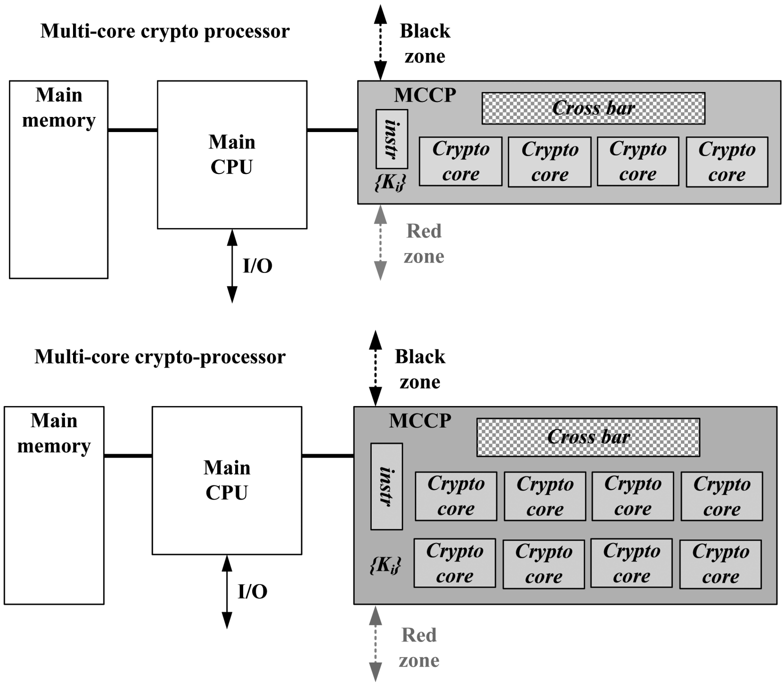


Fig. 8. Architectures of a four- and eight-core crypto processor.

presented is the routing topology used to exchange data between the processing element and the memory element (the memory bank). It is hard to estimate the best topology and architecture. Although some tools enable space exploration and evaluation, they usually do not give a sufficiently precise estimation of resource use and it is consequently very hard to compare different architectures fairly and accurately [Bossuet et al. 2007].

4. MULTICORE CRYPTO PROCESSOR

We have seen that crypto array architectures are efficient because they exploit the parallelism of the AES algorithm. In return, they are closely linked to this algorithm. The use of a crypto processor is more flexible because it is algorithm independent. Nevertheless, because of the sequential architecture of such crypto engines, their calculation performance is limited.

It is thus worthwhile considering the parallel implementation of several cryptographic computations (for one or more algorithms) by using multiple crypto processors in parallel. Such a solution resembles a crypto array structure with large processing elements. Each element is an elementary crypto processor (crypto core). This solution is illustrated by two architectures in Figure 8. It shows multicore crypto processors featuring four and eight crypto cores. Depending on the number of cores, the cores can communicate via a cross bar or via a NoC (Network on a Chip) in the case of a large number of crypto cores.

One example of a multicore crypto processor, the MCCP [Grand et al. 2011], focuses on designing a crypto processor, which has to provide the cryptographic services (e.g., secure SCA [Martin et al. 2008]) required by a secure SDR (Software-Defined Radio) base station. For this purpose, the crypto processor is embedded in a much larger component [Grand et al. 2009], which is placed at the boundary between the red part of

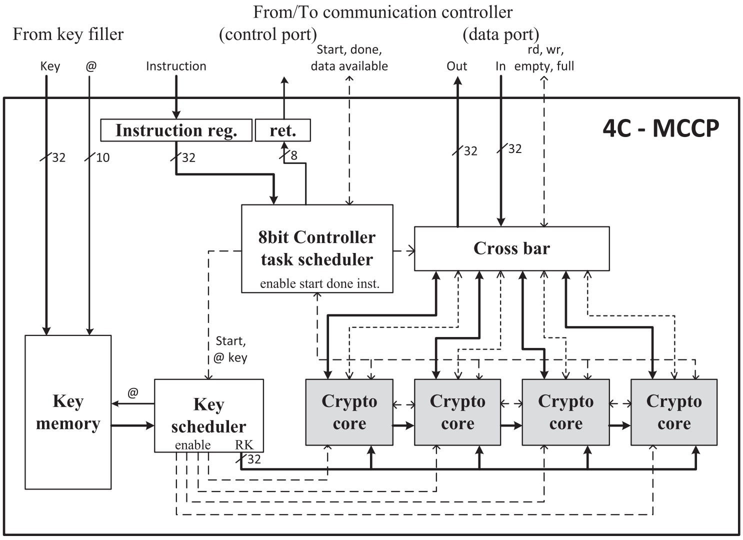


Fig. 9. Four-core MCCP architecture [Grand et al. 2011].

the radio (that processes plaintext data) and the black part of the radio (that processes encrypted data and sends/receives data to/from the RF component).

The *MCCP* [Grand et al. 2011] was designed considering that an SDR base station has to process several channels at the same time. The architecture proposed embeds several cryptographic cores to enable flexible and efficient enciphering/deciphering of the channel. MCCP architecture is scalable; up to eight cryptographic cores can be implemented. A four-core implementation of the MCCP is illustrated in Figure 9. In addition, the proposed design aims to make the FPGA platform as flexible as software components embedded in an SDR. Finally, for security reasons, the MCCP does not generate session keys itself; the keys are generated by its main controller and stored in a dedicated key memory.

The MCCP embeds a task scheduler that distributes cryptographic tasks to crypto cores. Task scheduler implementation uses a simple 8-bit controller that executes the task scheduling software. It manages the key scheduler, the cross bar, and the crypto cores. The task scheduler receives its orders from a 32-bit instruction register and returns values to the communication controller through the 8-bit return register (refer to Figure 8.).

Each crypto core communicates with the communication controller through the cross bar, which enables the task scheduler to select a specific core for I/O access. The key scheduler generates round keys from the session key stored in the key memory. Before launching key scheduling, the task scheduler loads the session key ID into the key scheduler which fetches the right session key from the key memory. To improve system security, the key memory cannot be accessed in write mode by the MCCP. In addition, there is no way to access the secret session key directly from the MCCP data port.

5. COMPARATIVE STUDY

The last part of Section 2 summarizes the main architectures of published crypto engines. It is very difficult and in fact usually impossible to objectively compare their performance. Recently, Kris Gaj et al. pointed out that no clear and commonly accepted method exists for comparing hardware performance of cryptographic algorithms implementation [Gaj et al. 2010]. The majority of reported results/evaluations were

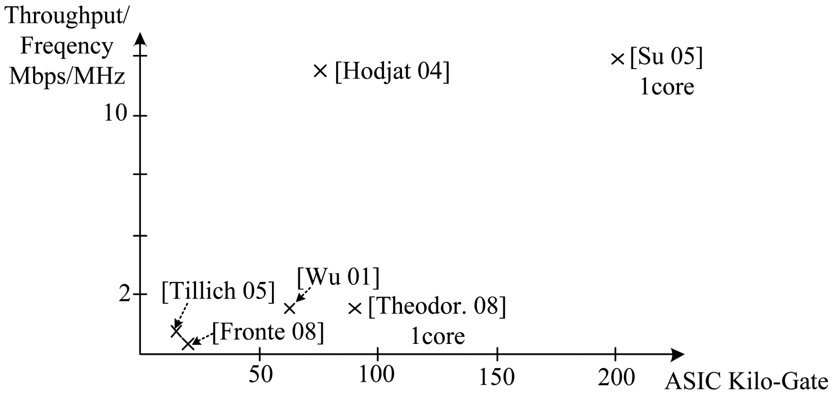


Fig. 10. Area vs. throughput/MHz for ASIC implementation.

performed on an ad hoc basis, and focused on one particular technology and one particular hardware family. Tools like ATHENa, an automated tool for hardware evaluation presented in Gaj et al. [2010], help compare several algorithms implemented in the same hardware. However, the tool is not able to compare different implementations of the same algorithm. For this reason, the main objective of our comparative study is not to give an accurate performance evaluation but to provide as much information as possible in order to compare performance in the most objective way possible. Table I lists some implementation results for the crypto engines described here. Interested readers should refer to the original papers that presented the architecture to familiarize themselves with all implementation results and with the setups (hardware target, metrics, and technology). Most results were obtained for the AES cipher in several cipher modes (ECB, CBC, CCM, and GCM). Only *CryptoBooster* implementation results were obtained for the IDEA cipher, probably due to the early year of publication (1999).

Some of the previously described works were implemented in ASIC technologies (from 0.25 μm CMOS technology to 32 nm). We depict these works in a 2D graph (Figure 10) with the occupied area (in kilo-gates) on the x-axis, and the throughput normalized per clock frequency (in mega-bits-per-second-per-mega-hertz) on the y-axis.

The other designs were implemented in FPGA devices. All these works used Xilinx devices: Virtex, Virtex-II, Virtex-II Pro, Virtex 4, and Virtex 6. The results are presented in Table II. The logic occupation is given in slices. For Virtex, Virtex-II, Virtex-II Pro and Virtex 4 a slice (called V-Slice) is mainly composed of two 4-input, 1-output Look-Up, Tables (LUTs) and two flip-flops. A Virtex 6 slice is composed of four 5-input, 2-output LUTs and eight flip-flops. As a consequence, one Virtex-6 slice (called V6-Slice) is equal to four V-Slices. Indeed, the number of RAM bits in V-Slice LUTs is 32 bits ($2^*2^{4*}1$ bits) and the number of RAM bits in a V6-Slice is 256 bits ($4^*2^{5*}2$ bits). The number of flip-flops in a V6-Slice is four times the number of those in a V-Slice. For this reason, Table I gives the number of equivalent V-Slices. The only difference concerns HCrypt [Gaspar et al. 2010], which uses Virtex-6 FPGA (denoted (*) in Table I). Figure 11 presents the published works in a 2D graph. The number of FPGA V-Slices is given on the x-axis (at a logarithmic scale), and the throughput in mega-bits-per-second-per-mega-hertz on the y-axis. Because the *CryptoBooster* uses different enciphering algorithms (IDEA), this work is not included in Figure 11.

As already mentioned, implementation results are hard to compare. Figure 10 and Figure 11 show the relative location of different works on a 2D graph. Some of them, Hodjat et al. [2004a, 2004b] for Figure 10, and Pericàs et al. [2008] and Su et al. [2005] for Figure 11, are a very good trade-off between area/resources use and performance

Table I. Summary of Results of Crypto Engine Implementation

Project architecture	Name [ref]	Silicon target	Algo. (mode)	Mbps/MHz	Max Freq.MHz	FPGA V-Slice	FPGA RAM	ASIC gates area
Custom GPP	Custom Xtensa [Ravi 02]	*Xtensa 0.18 μm*	*AES-ECB*	*0.11*	*188*	-	-	-
	Sbox instruction [Burke 00]	*Alpha 21264*	*AES-ECB*	0.05	600	-	-	-
	Inst. set ext. [Tillich 05]	*ASIC 0.13 μm*	*AES-ECB*	*0.57*	*250*	-	-	*16 KG* $0.08\ mm^2$
	CryptoBlaze [Xilinx 03]	*CPLD* *CoolRunner*	*AES-ECB*	*?*	*?*	-	-	-
	Intel AES inst. [Gueron 10]	*In tel core* *32 nm*	*AES-ECB*	*0.78*	*2670*	-	-	-
Crypto coprocessor	AES Processor [Hodjat 04a,b]	*ASIC 0.18 μm*	*AES-ECB*	*11.60*	*295*	-	-	*73 KG* $0.73\ mm^2$
	CryptoBooster [Mosanya 99]	*FPGA XCV1000*	*IDEA-ECB*	*16*	*33*	*?*	*?*	-
	AESTHETIC 1 core [Su 05]	*ASIC 0.25 μm*	*AES-ECB* *AES-CBC*	*12.80*	*66*	-	-	*200KG* $6.29\ mm^2$
	AESTHETIC 3 cores [Su 05]	*FPGA* *XCV2V6000*	*AES-ECB* *AES-CBC*	*36.80*	*50*	*27,561*	*0*	-
	CrCU [Chaves 06]	*FPGA* *XCV2VP30*	*AES-ECB*	*0.59*	*100*	*847*	*216 KB*	-
	AES-MS 2 cores [Pericàs 08]	*FPGA* *XCV2VP30*	*AES-ECB* *AES-CBC*	*25.60*	*100*	*2,161*	*432KB*	-
	SANES [Gognia t 08]	*FPGA* *XCV2VP30*	*AES-ECB*	*10.60*	*37.8*	*2,192*	*0*	-
Crypto processor *one-core and multi-core*	CryptoManiac [Wu 01]	*ASIC 0.25μm*	*AES-ECB*	*1.42*	*360*	-	-	$1.93\ mm^2$
	Cryptonite [Butchy 04]	*FPGA* *XCV2P30*	*AES-ECB*	*5.62*	*100*	*1,748*	*32KB.*	-
	CCProc 4 cores [Theodor. 08]	*FPGA* *XCV4LX200*	*AES-ECB*	*6.40*	*95*	*18,045*	*?*	-
	CCProc 1 core [Theodor. 08]	*ASIC 0.13 μm*	*AES-ECB*	*1.62*	*250*	-	-	*93KG* $5.3\ mm^2$
	HCrypt [Gaspar 10]	*FPGA XCV6LX*	*AES-ECB*	*1.60*	*150*	*7,960(*)*	*9.75 MB*	-
	MCCP [Grand 11]	*FPGA* *XCV4SX35*	*AES-CCM* *AES-GCM*	*4.43* *9.91*	*192*	*8110*	*7 MB*	-
	Celator [Fronte 08]	*ASIC 0.13 μm*	*AES-CBC* *SHA-256*	*0.24* *0.19*	*190*	-	-	*20 KG* $0.1\ mm^2$
Crypto array	CryptArray [Lomonaco 04]	*FPGA* *XCV2VP125*	*AES-ECB*	*1.27*	*81*	*>50,000*	*0*	-
	COBRA [Elbirt 03]	*FPGA XCV1000*	*AES-ECB*	*1.40*	*102*	*>10,000*	*0*	-

Table II. Summary of Crypto Engine Characteristics

Project architecture	Name [ref]	YoP*	Supported algorithm	Processing architecture	Key storage	Number of cryp to core	DHR**	Main application
Custom GPP	Custom Xtensa [Ravi 02]	*2002*	*DES, 3DES, AES, RSA*	*32-Bit Xtensa RISC processor customized ALU*	*In main memory*	*1 crypto ALU*	*no*	*Wireless data security*
	Sbox instruction [Burke 00]	*2000*	*3DES, IDEA, AES candidates*	*Sbox dedicated ALU*	*In main memory*	*1 Sbox look up table*	*no*	*IPSEC, VPN*
	Instruction set extension [Tillich 05]	*2005*	*AES*	*Custom SPARCV8 compatible 32-bit LEON-2*	*In main memory*	*1 Sbox MixColumns ShiftRows unit*	*no*	*Embedded system data security*
	CryptoBlaze [Xilinx 03]	*2003*	*AES,RSA*	*Custom 8-Bit Xilinx PicoBlaze*	*In main memory*	*Sbo x, GF multiplier n it*	*yes*	*FPGAs ystem data security*
	Intel AES inst. [Gueron 10]	*2010*	*AES*	*Intel IA-32*	*In main memory*	*1 AES ALU*	*no*	*PC client and server security*
Crypto coprocessor	AES processor [Hodjat 04]	*2004*	*AES-ECB, CBC-MAC, CCM*	*Custom SPARCV8 compatible 32-bit LEON-2*	*Embedded key register*	*1 full hardware AES engine*	*no*	*IPSEC, VPN*
	CryptoBooster [Mosanya 99]	*1999*	*IDEA, DES*	*Hardware reco nfigurable core*	*Sessionmemory*	*1 full hardware block cipher engine*	*yes*	*Network security*
	AESTHETIC [Su 05]	*2009*	*AES-ECB, CBC*	*Host pro cessor + hardware AES accelerator*	*Embedded key gen erator register*	*Single and multifull AES engines (1 to 3)*	*yes*	*Network security*
	CrCU [Chaves 06]	*2006*	*AES-ECB, AES-CBC, SHA-128, SHA-256*	*Molen processor + several AES cores*	*Embedded key register*	*CrCU number not limited*	*no*	*Trusted computing*
	AES-MS [Pericàs 08]	*2010*	*AES-ECB, AES-CBC*	*Molen processor + two AES cores*	*Embedded key register*	*2 AES cores*	*no*	*VPN*

(Continued)

Table II. (Continued)

Project architecture	Name [ref]	YoP*	Supported algorithm	Processing architecture	Key storage	Number of cryp to core	DHR**	Main application
Crypto processor *one-core and multi-core*	SANES [Gogniat 08]	*2005*	*AES, SHA . . . others*	*Reconfigurable hardware accelerator*	*Embedded key register*	*Up to 4 parallel crypto primitives*	*yes*	*Emb edded system security*
	CryptoManiac [Wu 01]	*2001*	*AES, DES, 3DES*	*4-wide 4 S tage VLIW processor*	*Internal shared data memory*	*4 crypto ALUs by processor*	*no*	*IPSEC, VPN*
	Cryptonite [Butchy 04]	*2004*	*AES, DES, MD5*	*2 *64-bit dedicated ALU*	*In main memory*	*Two dedicated ALUs*	*no*	*IPSEC, VPN*
	CCProc [Theodor 08]	*2008*	*AES candidates*	*VLIW based processor*	*In main memory*	*4 Sbox clusters*	*no*	*Symmetric encryption accelerator*
	HCrypt [Gaspar 10]	*2010*	*AES ECB, CFB, OFB, CTR*	*2 *32-bit dedicated ALU*	*In dedicated secured register*	*2 AES ciphers or deciphers*	*yes*	*Network security, VPN*
	MCCP [Grand 11]	*2011*	*AES, SHA . . . others*	*Multi-core processor*	*In dedicated register*	*From 2 to 8 reconfigurable crypto cores*	*yes*	*Software radio security*
Crypto array	Celator [Fronte 08]	*2008*	*AES, DES, 3DES, SHA-256*	*4*4 8-bit processing elements*	*In main memory*	*16 processing elements*	*yes*	*Network security*
	CryptArray [Lomonaco 04]	*2004*	*AES, DES, 3DES*	*Matrix of 4-bit processing elements*	*In main memory*	*Depend of a array size*	*yes*	*Network security*
	COBRA [Elbirt 03]	*2003*	*block ciphers*	*4*4 32-bit reconfigurable cryptographic element*	*In main memory*	*16 reconfigurable elements*	*yes*	*VPN*

*Year of Publication ** Design for hardware reconfiguration.

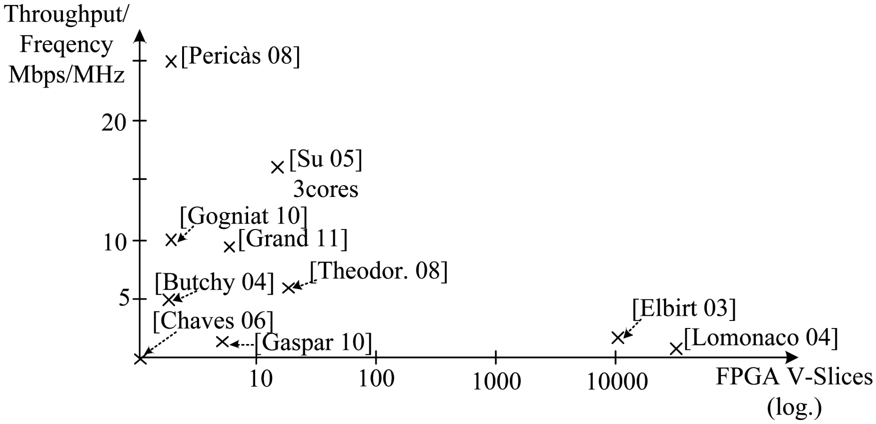


Fig. 11. Area vs. throughput/MHz for FPGA implementation.

(in terms of throughput). Nevertheless, the results presented in Table I, Figure 10, and Figure 11 are directly extracted from publications featuring specific hardware setups. Some results are given for a complete system (i.e., including the communication process, management of the system, etc.) and other results are only given for the cipher (often without taking key management into account). This means that a direct comparison of these results could lead to misinterpretation. We recommend that interested readers read the paper concerned to get more details about the context of experimental setup and interpretation of results.

To help the reader, Table II gives summarizes previously presented crypto engine features and capabilities. Most of the solutions in the table focus on data security applications such as Internet security (IPSec), Virtual Private Network (VPN), and secure embedded systems. The main application field of each proposal depends to a great extent on the year of publication. Each application field has specific constraints, for example, embedded system security has limited silicon resources and a limited power budget whereas communication network security requires a high processing rate to fully exploit the available network bandwidth. Other applications can be found in the literature such as e-voting [Neff 2011], Digital Rights Management (DRM) [Coburn et al. 2005], hardware-constrained sensor network node security [Roman et al. 2007], and software radio security [Grand et al. 2009].

It is interesting to note that in most of the previously presented architectures, secret key and/or session key secure storage were not considered as a security feature. That is to say, the most important part of the algorithm security was not taken into consideration in the design of the crypto engine. This lack of security could represent a significant failure in many works.

Concerning the technical development of crypto engines, the previously presented works and the works presented next cover a 12-year period (from 1999 to 2011), they are from distinct fields of application including microprocessor design, multicore processors, hardware design, FPGA implementation, and reconfigurable architecture (coarse grain). Table III lists, the works and the scientific field of each work (of course, the field of applied cryptography could appear in all works) in chronological order. In Table III, we can observe a trend in an increasing number of designers paying more attention to all design aspects, in order to meet the triple constraints of crypto engine design, as described in the Introduction of this article.

We can also observe that works which only aim at customization of a GPP with an ALU dedicated to cryptographic computations are less common today. This

Table III. Technical Scope of Each Work (in chronological order)

Name [ref]	Processor design	Multi-core processor	Hardware design	FPGA implementation	Reconfigurable architecture
CryptoBooster [Mosanya 99]	■		■	■	
Sbox instruction [Burke 00]	■				
CryptoManiac [Wu 01]	■	■			
Custom Xtensa [Ravi 02]	■				
CryptoBlaze [Xilinx 03]	■			■	
COBRA [Elbirt 03]			■		■
AES processor [Hodjat 04 a,b]	■		■		
Cryptonite [Butchy 04]	■		■		
CryptArray [Lomonaco 04]			■		■
AESTHETIC [Su 05]	■		■	■	
Instruction set extension [Tillich 05]	■				
CrCU [Chaves 06]	■		■		
CCProc [Theodor. 08]	■		■		
AES-MS [Pericàs 08]	■		■		
Celator [Fronte 08]			■		■
SANES [Gogniat 08]	■		■	■	■
Intel AES instruction set [Gueron 10]	■				
HCrypt [Gaspar 10]	■		■	■	
MCCP [Grand 11]	■	■	■	■	■

configuration has usually been replaced by a hardware accelerator, that is a crypto coprocessor (reconfigurable or not) or a crypto processor. Indeed, with the increased integration density available today, the silicon area is a less restrictive constraint than was the case at the beginning of the 2000s. Today, it is only advantageous to customize existing general-purpose processors for applications that are very restrictive in terms of area, such as RFID and sensor networks.

6. MAIN CRYPTO ENGINE DESIGN CHALLENGES

Crypto engine design faces a number of challenges that have led to the emergence of new trends. The purpose of the last section before the conclusion is to underline some of the most significant challenges. This section is not intended to be exhaustive but highlights some remaining problems and provides the reader with emerging research challenges.

Secure by design. As shown in the previous section, from the very beginning, crypto engine design was performance oriented. Paradoxically, in many cases, security features were not sufficiently taken into consideration. Most security features are only linked to side-channel attack countermeasures (dual precharge logic, masked implementation, etc.) and not to a secure by design process. However, architectural robustness can be significantly improved if security aspects are taken into account at all steps in the design process. The most effective improvement to secure design can be made if the security of the key path and the key storage (secret key, session, and/or master key) are considered in great detail. Table II clearly shows that most of architectures cited in the literature store confidential keys in an unsecure main data memory. This makes the system very vulnerable to software attacks. Securing the link between the general-purpose processor and the main memory is the first point to consider [Vaslin et al. 2006]. Another idea concerns the creation of a secure key bus embedded in the crypto engine, as proposed by Gaspar et al. [2010]. One important aspect the designer needs to understand is that security is not just a novel field of application. Security is a new design constraint.

MCryptoPSoC. Our review of crypto engine architectures showed that parallelism is an efficient approach in crypto engine implementations. Mixed with hardware reconfiguration properties, parallelism can ensure flexibility and high performance in cryptographic processing. Some recent applications, such as software radio, which manages many communication channels using a variety of cryptographic services, highlight such flexible and powerful architecture [Grand et al. 2011]. There is also a need for heterogeneous MPSoC (multi-processor system-on-a-chip) that can embed many crypto engines. In turn, these ideas may lead designers to consider the concept of MCryptoPSoC (multi-crypto-processor system-on-a-chip) to perform several data security applications inside one SoC. It is also important to consider resource sharing (such as memory resources and communication buses) in security applications linked with security requirements. Designers need to provide agile and high-performance cryptographic resources designed to execute many applications securely.

Virtualization. Despite such improvements, to be used efficiently, crypto processors and MCryptoPSoCs require high-level management and secure hardware virtualization. Hardware virtualization of security modules (such as TPM) is gaining importance in both academia and industry because it is an alternative way to increase the overall use of hardware resources. Some very recent and interesting works highlight this issue [Biedermann et al. 2011; Cotret et al. 2011; Gaber and Pailles 2010].

Malicious hardware. Emerging works in the crypto engine design field concern protection against malicious hardware. Hidden backdoors and hardware Trojans are not a “creation of the mind”. They are embedded in real circuits and can lead to serious security failures [Karri et al. 2010; Tehranipoor and Koushanfar 2010]. Due to this recent problem, trusted hardware, IC authentication and trusted design schemes need to be developed. For example, a number of authors are currently studying the design of Physical Unclonable Functions (PUF) to be used as a circuit fingerprint [Gassend et al. 2002].

Security metrics. As we mentioned at the beginning of this article, it is difficult to measure or even estimate the security of a hardware cryptographic module. Some international standards, such as FIPS 140-2, help the designer choose the desired level of security before designing the hardware module. Nevertheless, no accurate metrics are available to measure the real postdesign security of the hardware module. Such metrics may be available for certain parts of the security system. For example, methods of evaluation for physical random number generators, such as the German AIS31, enable estimation of generator entropy per random number and thus ensure at least a theoretical level of security. New security evaluation methods will be needed in the future for validation of the hardware security module to conform to security requirements, for example, how to measure the resistance to side-channel attack(s) [Standaert 2011; Batina et al. 2011]. This is an open question: is the use of a formal proof possible? How can security metrics be upgraded? How accurate will the results be?

Standardization. Some problems the *security metrics* can be solved by standardization. As mentioned earlier in connection with current standards for hardware cryptography (e.g., FIPS/AIS), the standards are not intended for problem solving. Instead, standardization can assist designers. The evolution and democratization of crypto processors not only led to strong needs in this area but also widened the application space.

Test and security. Integrated circuit testing at the different manufacturing and lifecycle stages is used to find fault-free devices, to improve production yield by analyzing the cause of defects, and to prevent runtime failures. However, test features are intuitively not well-suited for integrity and confidentiality assurance since they facilitate controllability and observability of internal data. Several scan-based attacks on hardware implementations of DES and AES have been demonstrated [Mukhopadhyay et al. 2005; Bo et al. 2006], showing that test facilities are an ideal starting point for identification of relevant internal nodes and for retrieving confidential information. Several test scan attack countermeasures have been published, but a universal minimum-cost solution for protection against test security threats [Hely et al. 2011] does not exist.

Embedded system security. An embedded system is a complex and often highly heterogeneous system. It is mainly composed of programmable parts (microprocessors), internal communication systems (bus, network on chip), memory (instructions, data, configurations), control units, inputs and outputs, hardware (reconfigurable or not), and various peripherals (e.g., external communication). As a result, since the early 2000s, many academic works have shown the extent of the threat model of these systems. Indeed, the embedded systems threat model includes software security, hardware security, data security, intellectual properties security and network/communication security for communicating embedded systems [Ravi et al. 2004; Koopman 2004; Bossuet and Gogniat 2010]. The development of secure embedded systems is highly constrained and prevents direct reuse of the security XE "security" solutions (software and hardware) developed for other purposes (chip cards, desktop and laptop computers, and servers). It is thus vital to develop solutions tailored to embedded systems to fit their specific characteristics and in conjunction with development constraints. A combined software and hardware attack on the system and its communication channel can have serious repercussions for the security XE "security" of the system and its data, and is therefore a serious issue for which new solutions are needed in a large software and hardware design space. Crypto engines are included is this large design space and may represent a possible solution. Since exploring the large design space of embedded systems is a challenging task, many application fields consider embedded systems security as a hot topic (e.g., automotive embedded system [Checkoway et al. 2011; Koscher et al. 2010]).

Availability. It is of paramount importance not only to provide cryptographic functionality but also to design systems that can still provide cryptographic services while under attack. This availability dimension needs to be addressed during the design process in order to include different levels of countermeasures depending on availability requirements. Embedding sensors and monitors in the design allows the behavior of the system to be tracked. A security policy manager can provide the right level of availability in line with the user's priorities. This feature will become a major expectation as users require a high Quality-of-Service (QoS). Availability and QoS will be very important for crypto engines.

Post-quantum crypto engine. Cryptography is not a static research field. Although robust cryptographic algorithms such as public key cryptography (e.g., RSA, ECC) are available and secure today, progress in computer science could call current cryptographic system security into question. For example, quantum computers will be able to break most popular public key cryptographic systems. In order to prevent attacks by quantum computers, many authors focused on the so-called post-quantum cryptography [Bernstein et al. 2008]. Lattice-based cryptography, multivariable public key cryptography, and code-based cryptography are possible solutions to post-quantum cryptography. Nevertheless, post-quantum cryptographic systems are hard to implement. For example, the public key length used for code-based cryptography is too large, even though it has already been reduced from several hundred thousand bits to only several tens of thousands of bits [Cayrel et al. 2011]. Such key lengths could lead to inefficient hardware implementation. Many studies will be needed to design an efficient post-quantum crypto engine to be prepared for the emergence of quantum computers.

In this section, we presented some of the main challenges in the crypto engine design field. The cryptographic hardware engineering field includes a wide range of works: side-channel countermeasure design [Popp et al. 2007], efficient hardware implementation of new cryptographic algorithms such as homomorphic encryption [Gentry and Halevi 2011; Naehrig et al. 2011], lightweight device security (such as RIFD, smart cards) [Rolfes et al. 2008; Lin et al. 2010; Feller et al. 2011], TRNG (True Random Number Generator) design and characterization [Valtchanov et al. 2010], chip identification circuits in new CMOS technologies, passive and active IC metering [Maes et al. 2009; Baumgarten et al. 2010], etc.

7. CONCLUSION

The crypto engine design field currently resembles an architectural jungle. In this article, a quick walk through the jungle enabled us to propose crypto engine taxonomy and a state-of-the-art. We have presented some of the most recent research projects to illustrate some of new trends in this field: securing the link between the processor and memory, securing the internal key path of the crypto engine using a trust-by-design approach, and improving performance and flexibility by designing reconfigurable parallel architectures.

A review of ten years of research in the crypto engine design field showed that the architectures proposed were often determined by the target application field. We can assume that the next step will be the design of a highly parallel crypto engine for both secret key and public key applications. The number of crypto cores could be huge in order to reply to the growing demand for cryptography services required by new applications such as cognitive radio and cloud computing. However, to reduce hardware costs, the security modules inside the system need to be able to share hardware resources, while guaranteeing the security of the system. This requirement is very often in opposition with the need for clear physical separation of security modules. Meanwhile, new challenges are emerging in the design of very low-power, low-cost, and low-area crypto engines for smart dust and sensor networks. To conclude research in the general field of cryptographic engineering requires a lot of research and new original concepts.

REFERENCES

ALTERA 2011. Cyclone III fpga: Security. http://www.altera.com/products/devices/cyclone3/overview/security/cy3-security.html.

ARORA, D., RAGHUNATHAN, A., RAVI, S., SANKARADASS, M., JHA, N. K., AND CHAKRADHAR, S. T. 2006. Software architecture exploration for high-performance security processing on a multiprocessor mobile SoC. In *Proceedings of the 43rd Annual Design Automation Conference (DAC'06)*. ACM Press, New York, 496–501.

ANDERSON, R., BOND, M., CLULOW, J., AND SKOROBOGATOV, S. 2006. Cryptographic processors-a survey. *Proc. IEEE 94,* 2, 357–369.

ANDERSON, R. 2001. *Security Engineering. A Guide to Building Dependable Distributed Systems*. Wiley.

BADRIGNANS, B., DANGER, J.-L., FISCHER, V., AND GOGNIAT, G. 2011. *Security Trends for FPGAS: From Secured to Secure Reconfigurable Systems*. Springer.

BANGERTER, E., GULLASH, D., AND KRENN, S. 2011. Cache games-bringing access-based cache attacks on AES to practice. In *Proceedings of the 2nd International Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE'11).* 215–221.

BATINA, L., GIERLICHS, B., PROUFF, A., RIVAIN, M., STANDAERT, F.-X., AND VEYRAT-CHARVILLON, N. 2011. Mutual information analysis: A comprehensive study. *Springer J. Cryptol. 24*, 2, 269–291.

BAUMGARTEN, A., TYAGI, A., AND ZAMBRENO, J. 2010. Preventing IC piracy using reconfigurable logic barriers. *IEEE Des. Test* 27, 1, 66–75.

Bernstein D. 2005. Cache-timing attacks on aes. Res. rep. http://cr.yp.to/antiforgery/cachetiming-20050414.pdf.

Bernstein, D. J., Buchmann, J., and Dahmen, E. 2008. *Post-Quantum Cryptography*. Springer.

Biedermann, A., Stöttinger, M., Chen, L., and Huss, S. A. 2011. Secure virtualization within a multi-processor soft-core system-on-chip architecture. In *Proceedings of the 7th International Symposium on Applied reconfigurable Computing (ARC'11)*. Lecture Notes in Computer Science, vol. 6578, Springer, 385–396.

Bo, Y., Kaijie, W., and Karri, R. 2006. Secure scan: A design-for-test architecture for crypto chips. *IEEE Trans. Integr. Circ. Syst. 25*, 10, 2287–2293.

Bossuet, L. and Gogniat, G. Hardware security in embedded systems. In *Communicating Embedded Systems for Networks*, F. Krief, Ed., Wiley-ISTE.

Bossuet, L., Gogniat, G., and Philippe, J. L. 2007. Communication-oriented design space exploration for reconfigurable architectures. *EURASIP J. Embed. Syst. 2007*, 1, 1–20.

Bossuet, L., Gogniat, G., and Burleson, W. 2006. Dynamically configurable security for SRAM FPGA bistreams. *Indersci. Intern. J. Embed. Syst. 2006*, 2, 73–85.

Bossuet, L., Gogniat, G., and Philippe, J. L. 2005. Generic design space exploration for reconfigurable architectures. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05),* Vol. 04 .IEEE Computer Society, Los Alamitos, CA, 163–171.

Buchty, R., Heintze, N., and Oliva, D. 2004. Cryptonite – A programmable crypto processor architecture for high-bandwidth applications. In *Proceedings of the Organic and Pervasive Computing Conference (ARCS'04)*. Lecture Notes in Computer Science, vol. 2981, Springer, 184–198.

Burke, J., McDonald, J., and Austin, T. 2000. Architectural support for fast symmetric-key cryptography. In *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'00)*. ACM Press, New York, 178–189.

Castillo, J., Huerta, P., Mart, J. I. 2007. Secure IP downloading for sram fpgas. *Microprocess. Microsyst. 31*, 2, 77–86.

Cayrel, P. L., El Yousi Alaoui, S. M., Hoffman, G., Meziani, M., and Niebuhr, R. 2011. Recent progress in code-based cryptography. In *Proceedings of the International Conference on Information Security and Assurance (ISA'11)*. Springer, 21–32.

Chaves, R., Kuzmanov, G., Vassiliadis, S., and Sousa, L. A. 2006. Reconfigurable cryptographic processor. In *Proceedings of the Workshop on Circuits, Systems and Signal Processing (ProRisc'06)*.

Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., Savage, S., Koscher, K., Czeskis, A., Roesner, F., and Kohno, T. 2011. Comprehensive experimental analyses of automotive attack surfaces. In *Proceedings of the 20th USENIX Conference on Security*. 6.

Coburn, J., Ravi, S., Raghunathan, A., and Chakradhar, S. 2005. SECA: Security-enhanced communication architecture. In *Proceeding of International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES'05).* ACM Press, New York, 78–89.

Cook, D. L., Ioannidis, J., Keromytis, A. D., and Luck, J. 2005. Cryptographics: Secret key cryptography using graphics cards. In *Proceedings of the Cryptographer's Track at the RSA Conference (CT-RSA'05).* 334–350.

Cotret, P., Crenne, J., Gogniat, G., Diguet, J. P., Gaspar, L., and Duc, G. 2011. Distributed security for communications and memories in a multiprocessor architecture. In *Proceeding of 25th International Parallel and Distributed Processing Symposium (IPDPS'11)*. IEEE Computer Society, 321–324.

Davies, P. 2003. *Flexible Security. White Paper, Cryptography and Interoperability*. Thales.

Deguang, L., Jinyi, C., Xingd, G., Ankang, Z., and Conglan, L. 2010. Parallel aes algorithm for fast data encryption on gpu. In *Proceedings of 2nd International Conference on Computer Engineering and Technology (ICCET'10)*. Vol. 6. ASME, New York, 1–6.

Duc, G. and Keryell, R. 2006. CryptoPage: An efficient secure architecture with memory encryption, integrity and information leakage protection. In *Proceedings of the 22nd Annual Computer Security Applications Conference (ACSAC'06).* IEEE Computer Society, 483–492.

Duc, G. and Keryell, R. 2008. Improving virus protection with an efficient secure architecture with memory encryption, integrity and information leakage protection. *Comput. Virol. 4*, 2, 101–113.

Eisenbarth, T., Guneysu, T., Paar, C., Sadeghi, A. R., Wolf, M., and Tessier, R. 2007a. Establishing chain of trust in reconfigurable hardware. In *Proceedings of the 15th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'07).* IEEE Computer Society, Los Alamitos, CA, 289–290.

Eisenbarth, T., Guneysu, T., Paar, C., Sadeghi, Schellekens, D., and Wolf, M. 2007b. Reconfigurable trusted computing in hardware. In *Proceedings of the Workshop on Scalable Trusted Computing (STC'07).* ACM Press, New York, 15–20.

ELBAZ, R., TORRES, L., SASSATELLI, G., GUILLEMIN, P., AND BARDOUILLET, M. 2006. PE-ICE: Parallelized encryption and integrity checking engine. In *Proceedings of the 9th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems (DDECS'06).* IEEE Computer Society, Los Alamitos, CA, 141–142.

ELBIRT, A. J. AND PAAR, C. 2003. Instruction-level distributed processing for symmetric-key cryptography. In *Proceedings of the 17th International Parallel and Distributed Processing Symposium (IPDPS'03).* IEEE Computer Society, Los Alamitos, CA, 78–88.

FELLER, T., MALIPATLOLLA, S., MEISTER, D., AND HUSS, S. A. 2011. TyniTPM: A lightweight module aimed to ip protection and trusted embedded platforms. In *Proceedings of the International Symposium on Hardware Oriented Security and Trust (HOST'11).* 60–74.

FRONTE, D., PEREZ, A., AND PAYRAT, E. 2008. Celator: A multi-algorithm cryptographic co-processor. In *Proceedings of the International Conference on Reconfigurable Computing and FPGAs (ReConFig'08).* IEEE Computer Society, Los Alamitos, CA, 438–443.

GABER, C. AND PAILLES, J. C. 2010. Security and trust for mobile phones based on virtualization. In *Proceedings of the 3rd Norsk Information Security Conference (NISK'10).* 93–103.

GAJ, K., KAPS, J.-P., AMIRINENI, V., ROGAWSKI, M., HOMSIRIKAMOL, E., AND BREWSTER, B. Y. 2010. ATHENA – Automated tool for hardware evaluation: Toward fair and comprehensive benchmarking of cryptographic hardware using FPGAs. In *Proceedings of the 20th International Conference on Field Programmable Logic and Applications (FPL'10).* IEEE Computer Society, Los Alamitos, CA, 414–421.

GARCIA, P., COMPTON, K., SCHULTE, M., BLEM, E., AND FU, W. 2006. An overview of reconfigurable hardware in embedded systems. *EURASIP J. Embed. Syst. 2006*, 1, 1–19.

GASPAR, L., FISCHER, V., BOSSUET, L., AND FOUQUET, R. 2011. Secure extensions of soft core general-purpose processors for symmetric key cryptography. In *Proceedings of the 6th International Workshop on Reconfigurable Communication-Centric Systems-on-Chip (ReCoSoC'11).* IEEE CAS Society.

GASPAR, L., FISCHER, V., BERNARD, F., BOSSUET, L., AND COTRET, P. 2010. HCrypt: A novel reconfigurable cryptoprocessor with secured key management. In *Proceedings of the International Conference on Reconfigurable Computing and FPGAs (ReconFig'10).* IEEE Computer Society, Los Alamitos, CA, 280–285.

GASSEND, B., CLARKE, D., VAN DIJK, M., AND DEVADAS, S. 2002. Silicon physical random functions. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS'02).* ACM Press, New York, 148–160.

GENTRY, G. AND HALEVI, S. 2011. Implementing grentry's fully-homomorphic encryption scheme. In *Proceedings of the 30th Annual International Conference on Theory and Applications of Cryptographic Techniques: Advanced in Cryptology (EUROCRYPT'11).* K. G. Paterson, Ed., Springer, 129–148.

GUERON, S. 2010. Intel Advanced Encryption Standard (AES) Instructions Set. White paper, Intel Mobility group, Israel Development Center, Israel.

GLAS, B., KLIMM, A., SANDER, O., MÜLLER-GLASER, K., AND BECKER, J. 2008. A system architecture for reconfigurable trusted platforms. In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE'08).* ACM Press, New York, 541–544.

GOGNIAT, G., WOLF, T., BURLESON, W., DIGUET, J. P., BOSSUET, L., AND VASLIN, R. 2008. Reconfigurable hardware for high-security/high-performance embedded systems: The safes perspective. *IEEE Trans. VLSI Syst. 16*, 2, 144–155.

GRAND, M., BOSSUET, L., LE GAL, B., DALLET, D., AND GOGNIAT, G. 2009. A reconfigurable crypto sub system for the software communication architecture. In *Proceedings of the IEEE Military Communication Conference (MILCOM'09).* IEEE Press, 2708–2714.

GRAND, M., BOSSUET, L., LE GAL, B., GOGNIAT, G., AND DALLET, D. 2011. Design and implementation of a multicore crypto-processor for software defined radios. In *Proceedings of the 7th International Symposium on Applied Reconfigurable Computing (ARC'11).* Lecture Notes in Computer Science, vol. 6578, Springer, 29–40.

GUNEYSU, T., MOLLER, B., AND PAAR, C. 2007. Dynamic intellectual property protection for reconfigurable devices. In *Proceedings of the International Conference on Field-Programmable Technology (FPT'07).* IEEE Electron Devices Society, 169–176.

HALDERMAN, J. A., SCHOEN, S. D., HENINGER, N., CLARKSON, W., PAUL, W., ALANDRINO, J. A., FELDMAN, A. J., APPELBAUM, J., AND FELTEN, E. W. 2009. Lest we remember: Cold boot attacks on encryption keys. *Comm. ACM 52*, 91–98.

HÄMÄLÄINEN, P., HÄNNIKÄINEN, M., AND HÄMÄLÄINEN, T. 2007. Review of hardware architectures for advanced encryption standard implementations considering wireless sensor networks. In *Proceedings of the 7th International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS'07).* Lecture Notes in Computer Science, vol. 4599, Springer, 443–453.

HELY, D., ROSENFELD, K., AND KARRI, R. 2011. Security challenges during vlsi test. In *Proceedings of the 9th IEEE NEWCAS Conference*. 1–4.

Hodjat, A. and Verbauwhede, I. 2004a. High-throughput programmable cryptocoprocessor. *IEEE Micro. 34*, 3, 34–45.

Hodjat, A. and Verbauwhede, I. 2004b. Interfacing a high speed crypto accelerator to an embedded CPU. In *Proceedings of the 38th Asilomar Conference on Signals, Systems and Computers*. 488–492.

Hodjat, A. and Verbauwhede, I. 2006. Area-throughput trade-offs for fully pipelined 30 to 70 gbits/s aes processors. *IEEE Trans. Comput. 55*, 4, 366–372.

Hori, Y., Satoh, A., Sakane, H., and Toda, K. 2008. Bitstream encryption and authentication using aes-gcm in dynamically reconfigurable systems. In *Proceedings of the 3rd International Workshop on Security: Advances in Information and Computer Security (IWSEC'08)*. Springer, 261–278.

Kaps, J. P. and Paar, C. 1998. Fast des implementation for fpgas and its application to a universal key-search machine. In *Proceedings of the 5th Annual International Workshop on Selected Areas in Cryptography (SAC'98)*. S. E. Tavares and H. Meijer, Eds., Springer, 234–247.

Karri, R., Rajendran, J., Rosenfeld, K., and Tehranipoor, M. 2010. Trustworthy hardware: Identifying and classifying hardware trojans. *Comput. 43*, 10, 39–46.

Koopman, P. 2004. Embedded system security. *Comput. 37*, 7, 95–97.

Koscher, K., Czeskis, A., Roesner, F., Patel, S., Kohno, T., Chekoway, S., Mccoy, D., Kantor, B., Aderson, D., Shacham, H., and Savage, S. 2010. Experimental security analysis of a modern automobile. In *Proceedings of the IEEE Symposium on Security and Privacy*. 447–462.

Kuzmanov, G., Gaydajiev, G. N., and Vassiliadis, S. 2004. The molen processor prototype. In *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'04)*. 296–299.

Lee, R. B., Kwan, P. C. S., Mcgregoc, J. P., Dwoskin, J., and Wang, Z. 2005. Architecture for protecting critical secrets in microprocessors. In *Proceedings of the 32nd International Symposium on Computer Architecture (ISCA'05)*. IEEE Computer Society, Los Alamitos, CA, 2–13.

Lie, D., Thekkath C., Mitchell. M., Lincoln, P., Boneh, D., Mitchell, J., and Horowitz, M. 2000. Architectural support for copy and tamper resistant software. In *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'09)*. 168–177.

Lin, L., Holcomb, D., Kumar Krishnappa, D., Shabadi, P., and Burleson, W. 2010. Low-power subthreshold design of secure physical unclonable functions. In *Proceedings of the 16th ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED'10)*. ACM Press, New York, 43–48.

Lomonaco, M. 2004. Cryptarray a scalable and reconfigurable architecture for cryptographic applications. Masters thesis, University of Central Florida.

Manavski, S. A. 2007. CUDA compatible gpu as an efficient hardware accelerator for aes cryptography. In *Proceedings of International Conference on Signal Processing and Communications (ICSPC'07)*. IEEE, 65–68.

Martin, A., Newman, T., and Morotake, D. 2008. Development approaches for an international tactical radio cryptographic api. In *Proceedings of the Software Design Radio Technical Conference* (*SDRForum'08)*. 1–6.

Maes, R., Schellekens, D., Tuyls, P., and Verbauwhede, I. 2009. Analysis and design of active IC metering schemes. In *Proceedings of the IEEE International Workshop on Hardware-Oriented Security and Trust (HOST'09)*. IEEE Computer Society, Los Alamitos, CA, 74–81.

Malipatlolla, S. and Huss, S. A. 2011. A novel method for secure intellectual property deployment in embedded systems. In *Proceeding of 7th Southern International Conference on Programmable Logic (SPL'11)*. IEEE Circuits and Systems Society, 1–6.

Mosanya, E., Teuscher, C., Restrepo, H. F., Galley, P., and Sanchez, E. 1999. CryptoBooster: A reconfigurable and modular cryptographic coprocessor. In *Proceedings of the 1st International Workshop on Cryptographic Hardware and Embedded Systems (CHES'99)*. Lecture Notes in Computer Science, vol. 1717, Springer, 246–257.

Morabi, A., Barenghi, A., Kasper, T., and Paar, C. 2011. On the vulnerability of FPGA bitstream encryption against power analysis attacks: extracting keys from xilinx virtex-II FPGAs. In *Proceedings of the 18th ACM Conference on Computer and Communication Security (CCS'11)*. ACM Press, New York, NY, 111–124.

Morabi, A., Kasper, M., and Paar, C. 2012. Black-box side channel attacks highlight the importance of countermeasures – An analysis of the xilinx virtex-4 and virtex-5 bitstream encryption mechanism. In *Topics in Cryptology: The Cryptographer's Track at the RSA Conference (CT-RSA'12)* (To appear).

Mucci, C., Vanzolini, L., Campi, F., and Toma, M. 2007. Interactive presentation: Implementation of aes/rijndael on a dynamically reconfigurable architecture. In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE'07)*. ACM Press, New York, 355–360.

Mukhopadhyay, D., Banerjee, S., Roychowdhury, D., and Bhattacharya, B. B. 2005. CryptoScan: A secured scan chain architecture. In *Proceedings of the 14th Asian Test Symposium (ATS'05)*. 348–343.

NAEHRIG, M., LAUTER, K., AND VAILKUNTANATHAN, V. 2011. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM Workshop on Cloud Computing Security (CCSW'11)*. ACM Press, New York, 113–124.

NAKANISHI, M. 2008. An FPGA configuration scheme for bitstream protection. In *Proceedings of the 4th International Workshop on Reconfigurable Computing: Architectures, Tools and Applications (ARC'08)*. Springer, 330–335.

NEFF, C. 2011. A verifiable secret shuffle and its application to e-voting. In *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS'10)*. P. Samarati, Ed., ACM Press, New York, 116–125.

OSVIK, D. A., SHAMIR, A., AND TROMER, E. 2006. Cache attacks and countermeasures: The case of aes. In *Proceedings of the Cryptographers' Track at the RSA Conference (Ct-RSA'06)*. Lecture Notes in Computer Science, vol. 3860, Springer, 1–20.

PERICÀS, M., CHAVES, R., GAYDADJIEV, G. N., VASSILIADIS, S., AND VALERO, M. 2008. vectorized aes core for high-throughput secure environments. In *Proceedings of 8th International Meeting High Performance Computing for Computational Science (VECPAR'08)*. 83–94.

POPP, T., MANGARD, S., AND OSWALD, E. 2007. Power analysis attacks and countermeasures. *IEEE Des. Test 24,* 6, 535–543.

RAVI, S., RAGHUNATHAN, A., KOCHER, P., AND HATTANGADY, S. 2004. Security in embedded systems: Design challenges. *ACM Trans. Embed. Comput. Syst. 3,* 3, 461–491.

RAVI, S., RAGHUNATHAN, A., POTLAPALLY, N., AND SANKARDASS, M. 2002. System design methodologies for a wireless security processing platform. In *Proceedings of the 39th Annual Design Automation Conference (DAC'02)*. ACM Press, New York, 777–782.

REBEIRO, C., MUKHOPADHYAY, D., TAKAHASHI, J., AND FUKUNAGA, T. 2009. Cache timing attacks on clefia. In *Proceedings of 10th International Conference on Cryptology in India: Progress in Cryptology (Indocrypt'09)*. B. Roy and N. Sendrier, Eds., Springer, 104–118.

REBEIRO, C. AND MUKHOPADHYAY, D. 2011. Cryptanalysis of clefia using differential methods with cache trace patterns. In *Proceedings the Cryptographers' Track at the RSA Conference (CT-RSA'11)*. Lecture Notes in Computer Science, vol. 6558, Springer, 89–105.

REGAZZONI, F., EISENBARTH, T., BREVEGLIERI, L., IENNE, P., AND KOREN, I. 2008. Can knowledge regarding the presence of countermeasures against fault attacks simplify power attacks on cryptographic devices? In *Proceedings of the IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems (DFT'08)*. IEEE Computer Society, Los Alamitos, CA, 202–210.

ROLFES, C., POSCHMANN, A, LEANDER, G., AND PAAR, C. 2008. Ultra-lightweight implementations for smart devices - Security for 1000 gate equivalents. In *Proceedings of the 8th IFIP WG 8.8/11.2 International Conference on Smart Card Research and Advanced Applications (CARDIS'08)*. Lecture Notes in Computer Science, vol. 5189, Springer, 89–103.

ROMAN, R., ALCARAZ, C., AND LOPEZ, J. 2007. A survey of cryptographic primitives and implementations for hardware-constrained sensor network nodes. *Mob. Netw. Appl. 12,* 4, 231–244.

SAKIYAMA, K., BATINA, L., PRENEEL, B., AND VERBAUWHEDE, I. 2007a. HW/SW co-design for public-key cryptosystems on the 8051 micro-controller. *Comput. Electron. Engin. 33*, 5–6, 324–332.

SCHAUMONT, P. AND VERBAUWHEDE, I. 2003. Domain-specific codesign for embedded security. *Comput. 36*, 4, 68–74.

STANDAERT, F.-X. 2011. Some hints on the evaluation metrics and tools for side-channel attacks. In *Proceedings of the Non-Invasive Attacks Testing Workshop (NIAT'11).* http://perso.uclouvain.be/fstandae/PUBLIS/107_slides.pdf.

STANDAERT, F., VAN OLDENEEL TOT OLDENZEEL, L., SAMYDE, D., AND QUISQUATER, J. 2003. Power analysis of FPGAs: How practical is the attack? In *Proceedings of the 13th International Conference on Field Programmable Logic and Application (FPL'03)*. Lecture Notes in Computer Science, vol. 2778, Springer, 701–711.

SU, C. P., HORNG, C. L., HUANG, C. T., AND WU, C. W. 2005. A configurable aes processor for enhanced security. In *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC'05)*. ACM Press, New York, 361–366.

SUH, G. E., CLARKE, D., GASSEND, B., VAN DIJK, M., AND DEVADAS, S. 2003. AEGIS: Architecture for tamper-evident and tamper-resistant processing. MIT, Memo-461.

TCPA – TRUSTED COMPUTING PLATFORM ALLIANCE. 2003. TPM main specification version 1.1b. Trusted Computing Group.

TEHRANIPOOR, M. AND KOUSHANFAR, F. 2010. A survey of hardware trojan taxonomy and detection. *IEEE Des. Test* 27, 1, 10–25.

THEODOROPOULOS, D., PAPAEFSTATHIOU, I., AND PNEVMATIKATOS, D. N. 2008. CCproc: An efficient cryptographic coprocessor. In *Proceedings of 16th IFIP/IEEE International Conference on Very Large Scale Integration (VLSI'08)*. 160–163.

THEODOROPOULOS, D., SISKOS, A., AND PNEVMATIKATOS, D. N. 2009. CCproc: A custom vliw cryptography coprocessor for symmetric-key ciphers. In *Proceedings of the 5th International Workshop on Applied Reconfigurable Computing (ARC'09)*. Lecture Notes in Computer Science, vol. 5453, Springer, 318–323.

TILLICH, S., GROSSSCHÄDL, J., AND SZEKELY, A. 2005. An instruction set extension for fast and memory-efficient aes implementation. In *Proceedings of 9th International Conference on Communications and Multimedia Security (CMS'05)*. Lecture Notes in Computer Science, vol. 3677, Springer, 11–21.

TILLICH, S. AND GROSSSCHÄDL, J. 2006. Instruction set extensions for efficient aes implementation on 32-bit processors. In *Proceedings of the 8th International Conference on Cryptographic Hardware and Embedded Systems (CHES'06)*. Lecture Notes in Computer Science, vol. 4249, Springer, 270–284.

TILLICH, S. AND HERBST, C. 2008. Boosting aes performance on a tiny processor core. In *Proceedings of the Cryptopgraphers' Track at the RSA Conference on Topics in Cryptology (CT-RSA'08)*. Lecture Notes in Computer Science, vol. 4964, Springer, 170–186.

TIRI, K. AND VERBAUWHEDE, I. 2005. A vlsi design flow for secure side-channel attack resistant ics. In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE'05)*. Vol. 3, IEEE Computer Society, Los Alamitos, CA, 58–63.

TREDENNICK, N. AND SHIMAMOTO, B. 2003. The rise of reconfigurable systems. In *Proceeding of the Engineering of Reconfigurable Systems and Application (ERSA'03)*.

VASLIN, R., GOGNIAT, G., AND DIGUET, J. P. 2006. Secure architecture in embedded systems: An overview. In *Proceedings of the Workshop on Reconfigurable Communication-Centric SoCs (ReCoSoc'06)*. 1–9.

VASLIN, R., GOGNIAT G., DIGUET, J. P., WANDELEY, E., TESSIER, R., AND BURLESON, W. 2007. Low latency solution for confidentiality and integrity checking in embedded systems with off-chip memory. In *Proceedings of the Workshop on Reconfigurable Communication-centric SoCs (ReCoSoc'07)*.146–153.

VALTCHANOV, B., FISCHER, V., AUBERT, A., AND BERNARD, F. 2010. Characterization of randomness sources in ring oscillator-based true random number generators in fpgas. In *Proceedings of the 13th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS'10)*. IEEE Computer Society, Los Alamitos, CA, 48–53.

VERBAUWHEDE, I., HOORNAERT, F., VANDEWALLE, J., AND DE MAN, H. 1991. ASIC cryptographical processor based on des. In *Proceedings of the IEEE European Event in ASIC Design (EUROASIC'91)*. 292–295.

WANG, M. Y., SU, C. P., HORNG, C. L., WU, C. W., AND HUANG, C. T. 2010. Single- and multi-core configurable aes architectures for flexible security. *IEEE Trans. VLSI Syst. 18*, 4, 541–552.

WEAVER C., KRISHNA, R., WU, L., AND AUSTIN, T. 2001. Application specific architectures: a recipe for fast, flexible and power efficient designs. In *Proceedings of the International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES'01)*. ACM Press, New York, 181–185.

WOLLINGER, T., GUAJARDO, J., AND PAAR, C. 2004. Security on FPGAs: State-of-the-art implementations and attacks. *ACM Trans. Embed. Comput. Syst. 3*, 3, 534–574.

WOLLINGER, T. AND PAAR, C. 2003. How secure are fpgas in cryptographic applications. In *Proceeding of 13th International Conference on Field-Programmable Logic and Applications (FPL'03)*. Lecture Notes of Computer Science, vol. 2778, Springer, 91–100.

WU, L., WEAVER, C., AND AUSTIN, T. 2001. CryptoManiac: A fast flexible architecture for secure communication. In *Proceedings of the 28th Aannual International Symposium on Computer Architecture (ISCA'01)*. IEEE Computer Society, Los Alamitos, CA, 110–119.

XILINX CORP. 2001. Virtex 2.5V field programmable gate arrays. Product specification DS003-1. http://www.xilinx.com/products/silicon-devices/fpga/.

XILINX CORP. 2003. CryptoBlaze: 8-bit security microcontroller. Application note, XAPP374. http://www.xilinx.com/support/documentation/application_notes/xapp374.pdf.

XILINX CORP. 2010. PicoBlaze 8-bit embedded microcontroller user guide for spartan-3, saprtan-6, virtex-5 and virtex-6 fpgas. User guide, UG 129. http://www.xilinx.com/products/intellectual-property/picoblaze.htm.

XILINX CORP. 2012. Virtex 7 series FPGAs overview. Advance product specification ds180. http://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf.

ZHUANG, X., ZHANG, T., LEE, H. H. S., AND PANDE, S. 2004. Hardware assisted control flow obfuscation for embedded processors. In *Proceedings of the International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES'04)*. ACM Press, New York, 292–302.

Received October 2011; revised February 2012; accepted June 2012