Reconfigurable Security Primitive for Embedded Systems

Guy Gogniat
LESTER Laboratory
University of South Brittany
Lorient, France
Email: guy.gogniat@univ-ubs.fr

Tilman Wolf
Department of Electrical and
Computer Engineering
University of Massachusetts
Amherst, MA 01003-9284 USA
Email: wolf@ecs.umass.edu

Wayne Burleson
Department of Electrical and
Computer Engineering
University of Massachusetts
Amherst, MA 01003-9284 USA
Email: burleson@ecs.umass.edu

***Abstract*— Embedded systems present significant security challenges due to their limited resources, power constraints and a variety of inherent vulnerabilities. In this paper, we propose a reconfigurable security primitive for secure embedded systems that leverages the capabilities of reconfigurable hardware to provide an efficient and flexible architectural support to both security standards and a range of attacks. This paper stresses design challenges for secure embedded systems and argues the case for reconfigurable architectural support for security. The reconfigurable security primitive is based on two main ideas: 1) an adaptable datapath, and 2) a hierarchy of controllers at the primitive and system level. The first controller manages the performance policy while the second one deals with the security policy. The AES cryptography algorithm has been considered to show the benefit of our approach compared to hardware and software solutions.**

I. Introduction

Embedded systems are expected to play an essential role in the future and today they are already spread in many electronic devices from low-end to high-end systems [1]. The vision of ubiquitous computing is becoming a reality, however there is a major bottleneck to its widespread adoption, the security issue is a serious problem and attacks against these systems are becoming more critical and sophisticated [2][3][4]. Confidentiality and privacy are major concerns for users and today nearly 52% of cell phones users and 47% of PDA users feel that security is the single largest concern preventing the adoption of mobile commerce [5]. New solutions have to be proposed in order to allow the definition of secure embedded systems. Current technologies are facing several challenges as stressed by Ravi et al. [1]. Architectures will have to meet high performance, energy efficiency, flexibility, tamper resistance and reliability to enable the vision of ubiquitous computing. Reconfigurable technologies can address these requirements and provide efficient security primitives. Their characteristics enable the system to prevent attacks or to react when attacks are detected while guarantying the required energy and computation efficiency.

In this paper we propose a reconfigurable security primitive which emphasizes high-security and high-performance. It is based on a reconfigurable datapath in order to dynamically adapt its architecture depending on the system and environment states. Such an approach enables the system to provide the right security barrier leading to a high-performance and security-efficient solution. Thus, energy, throughput and tamper resistance can be adapted to meet dynamic constraints.

The remainder of this paper is organized as follows. Section 2 reviews the main challenges to provide secure embedded systems and stresses the benefit of using reconfigurable architectures. Section 3 describes our proposition, a reconfigurable security primitive and highlights its flexibility. Section 4 demonstrates the efficiency of our approach dealing with the AES security primitive as a case study. Finally, section 5 concludes the paper and proposes some extensions to our work.

II. Secure embedded system design challenges

Designers are facing the critical task of guarantying the security of increasing more complex systems while dealing with very tight constraints. This new design metric for embedded systems is very hard and gathers many aspects and layers that are difficult to manage [2]. Several challenges have to be addressed to provide an efficient and reliable solution. These have been stressed by Ravi et al. in [1] and a rapid review is proposed below.

- The *processing gap* highlights that current embedded system architectures are not capable of keeping up with the computational demands of security processing.
- The *battery gap* emphasizes that the current energy consumption overheads of supporting security on battery-constrained embedded systems are very high.
- The *flexibility* stresses that an embedded system is often required to execute multiple and diverse security protocols and standards.
- The *tamper resistance* emphasizes that secure embedded systems are facing an increasing number of attacks from physical to software attacks. Side-channel attacks also represent an important threat for these systems.
- The *assurance gap* is related to reliability and stresses the fact that secure systems must continue to operate reliably despite attacks from intelligent adversaries who intentionally seek out undesirable failure modes.

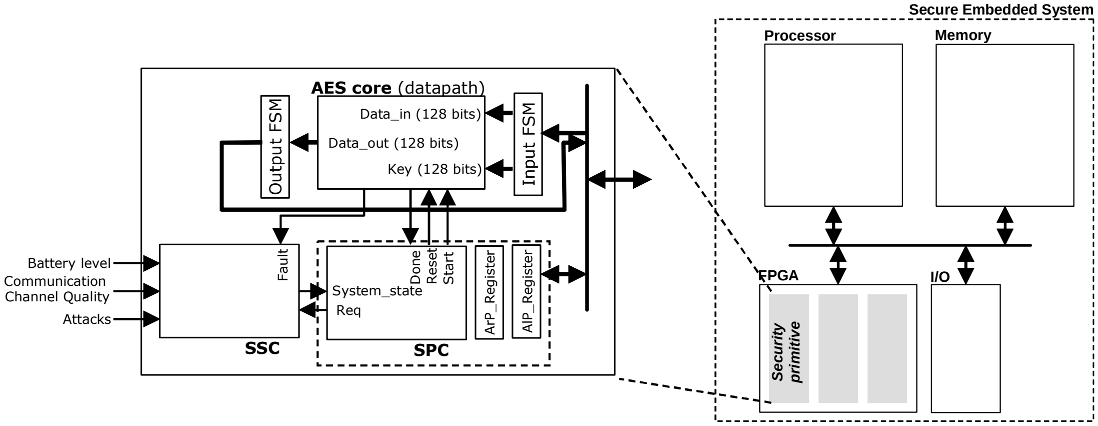


Fig. 1. Integration of the reconfigurable hardware within the embedded system architecture. The reconfigurable hardware provides the security primitives. The Security Primitive Controller manages the flexibility of the primitive and the Security System Controller deals with the detection of abnormal activity using specific sensors

Designing an embedded system architecture dealing with all these requirements is a challenging task, however reconfigurable technologies provide several features to mitigate these gaps, especially for security primitives. Several works have already been published dealing with security primitives and reconfigurable technologies, especially for cryptography algorithms. In [7] and [8] a thorough analysis of existing implementations for the AES candidates onto FPGAs is presented. However, the mechanisms required to manage the *flexibility* of these primitives is not considered. In [9] an adaptive cryptographic engine is proposed which enables the system to adapt the cryptography algorithm depending on the requirements. Though, the *reliability gap* is not addressed. In [10] the authors propose a thorough analysis of existing implementations of the AES security primitive dealing with both performance and reliability issues. However, previous efforts do not address all of the challenges stated by Ravi et al. [1]. Other solutions can be considered consisting of programmable hardware accelerators. In [11] and [12] the authors propose cryptography processor or co-processor which can perform various execution modes and achieve high throughput. However, they do not address the attack issue and the energy efficiency metric is not considered. Finally, in [13] the authors focus on architecture support for energy-efficient security. In their work they deal with security primitives and security protocols but they do not consider the attack issue.

The work presented in this paper differs from these efforts in several respects. First, we propose an architecture for security primitives which promotes the design of secure embedded systems by targeting all of the challenges stated by Ravi et al. [1]. Our architecture is based on a datapath and a hierarchy of controllers which enables dealing with flexibility and security. The performance and energy issues are considered by using a reconfigurable datapath which enables the system to provide several tradeoffs depending on the requirements and on the security policy. The reliability issue is managed through the use of different implementations from low to high reliability (e.g. fault detection or fault tolerance). The underlying concept of our approach is to dynamically adapt the security primitive in order to deal with dynamic constraints (i.e. attacks, performance, power). Such a solution enables the system to face an unsecured and evolving environment while meeting performance and constraints issues.

III. Reconfigurable security primitive

A. Toward a secure primitive

Before presenting our approach it is interesting to define what could be an *ideal* primitive. As stated before several features should be considered to provide an efficient solution. *Performance* and *energy* are closely related and have to be considered. Furthermore, it is mandatory to be able to dynamically adapt these features depending on the system state. *Flexibility* is another major feature as many standards are continuously evolving to face the increase pressure of hackers. *Tamper resistance* is becoming a major challenge, especially for hardware implementations as many side-channel attacks have been developed this last few years (e.g. timing, power and electromagnetic analysis). The primitive should be symptom-free so that providing any information (i.e. data leaks). Finally, *Reliability* has to be considered to guarantee the functionality of the system even in case of attacks (typically fault injection).

All these points raise several questions; What are the performance and the security policies? What should be the architecture of the security primitive? What are the overhead costs to provide flexibility and reliability? What are the links between the primitive and the system?

Answering all of these questions is a very complex task. However, we believe that it is important to separate the flexibility and the security management to provide a more reliable solution and to reduce the penalty costs of the security. Thus, a security primitive should be composed of a datapath and a hierarchy of controllers (flexibility and security controllers). The performance policy should be handle by one controller and should cope with various constraints to define when reconfiguring the datapath. Typical performance policies should be based on best effort and guaranteed throughput approaches. The security policy is more complex and must be defined through a defense-in-depth approach [14]. Depending on the attack several scenarios should to be considered as discarding some data or providing more secure architectures. In the following sections a detailed presentation of our approach is provided which stresses the role of each module within the primitive. Then, the result section will highlight the overhead costs to guarantee the flexibility of the system.

B. General overview of our primitive

The reconfigurable security primitive can be seen as an agile hardware accelerator which performs a security algorithm (e.g. cryptography, IP filtering, key management). An embedded system generally embeds several security primitives that can evolve during its lifetime (*flexibility gap*). Figure 1 highlights the links between the reconfigurable security primitives and the other modules within the embedded system. A Security Primitive Controller (SPC) is connected to the datapath in order to manage its flexibility. The SPC control tasks are related to reconfiguration of the datapath to change or adapt its architecture. The SPC is connected to the system processor in order to define the configuration of the security primitive. For example, in the case of cryptography it corresponds to the parameters of the algorithm (i.e. key size, mode and key value). A System Security Controller (SSC) is also connected to each security primitive to monitor the primitive and to check the system state to detect if some faults injection or abnormal operations are performed. The role of the SSC is to detect attacks against the system. The SSCs are connected to sensors to analyze the different parts of the system (e.g. battery, buses, other security primitives, communication channel).

C. Detailed architecture of the primitive

The reconfigurable security primitive is composed of the datapath and the two previous controllers (SPC and SSC) as shown in the left part of figure 1. The SPC is connected to the system processor through a memory mapped mechanism (i.e. hardware accelerator). Depending on the primitive, different configuration registers are used to define its configuration. These registers provide the algorithm (i.e. execution mode and key size for cryptography algorithm) and architecture parameters (i.e. throughput, area and reliability). As stated before, the SPC manages the flexibility of the primitive. When the processor needs a security primitive it first configures the SPC which starts to check what execution modes can be used. To define the algorithm and architecture parameters the SPC polls via the SSC the current state of the system. For example, battery level and communication channel quality sensors can be considered to define this state. Based on this information the SPC defines what type of parameters can be selected and provides these values to the processor.

Then, the processor selects the right parameters depending on its constraints and writes them into the configuration registers to control the SPC. The SPC performs the configuration of the datapath so that the primitive can be used. Once configured the datapath is independent and the SPC does not manage its I/Os. Such a solution allows the primitive to provide high performance. While the data are computed through the datapath, the SPC continues to poll via the SSC the state of the system to check if the mode of the security primitive needs to be changed (the aim is to change the mode if for example the battery is running low and the functionality is still to be available). If so, it computes the new architecture parameters and performs a reconfiguration of the datapath depending on the performance policy associated with the security primitive (i.e. best effort, guaranteed throughput). An interrupt to the processor is also generated to indicate that new architecture parameters have been implemented.

D. Protection against attacks

Two main scenarios have been considered in our work to protect the system from being pirated and to guarantee the execution of the security protections. The first one is managed by the SSC and the second one by the SPC. The SSC can interrupt the SPC if an irregular activity is detected within the module or the system. In that case the SSC indicates to the SPC what configuration has to be implemented. Example of attacks are hijacking, denial-of-service (e.g. draining of battery or causing battery to overheat) and extraction of secret information (e.g. user's phone book). In case of an hijacking attack the security primitive needs to be reconfigured with a secure configuration. In case of a denial-of-service attack the primitive needs to be enhanced by fault tolerance mechanisms to be able to guarantee its functionality and in case of an extraction of secret information attack, I/Os of the primitive need to be stalled.

Once the attack has been fended off the SPC defines a new configuration to provide the best performance tradeoff, for example in term of throughput versus energy when dealing with cryptography. Protected modes like fault tolerant architecture consume more area and power so it is essential to run these modes only when required and not by default to guarantee the power efficiency of the system. The security is also provided through the SPC since it continuously checks the state of the system to guaranty the best performance for the security primitive. Mobile devices are characterized by two main parameters, the power limitation and the evolving environment which leads to various level of quality of the communication channels. Hence, depending on both the SPC selects which parameters have to be considered. For example, in case of a best effort performance policy, when the level of battery is low or the channel quality decreases under some thresholds then the SPC reconfigures the module with a lower throughput but a better energy-efficient architecture. In case of guaranteed throughput, the SPC keeps the same parameters event if the thresholds are crossed.

The performance and security policies are essential to take benefit of the reconfigurability of the system and to provide efficient solutions. These policies are very dependent of the primitive and have to cope with its intrinsic specificities. In this study, we have defined very simple policies as the goal was mainly to demonstrate the advantages of such a solution instead of focusing on new policy algorithms. However, for interested readers, there are several papers that have been published about performance policy [15][16].

IV. AES RECONFIGURABLE SECURITY PRIMITIVE CASE STUDY

To illustrate the reconfigurable security primitive described in this paper an implementation of the AES algorithm is proposed in the following sections [17]. The AES algorithm as been selected for that study since it is expected to be one of the major cryptography algorithm in the future. Our implementation has been performed on a Xilinx FPGA Virtex-II Pro device (xc2vp30-5ff896) [18]. Figure 1 presents the reconfigurable security primitive architecture and the links between the processor and the memory that contains the different bitstreams (each bitstream corresponds to a configuration). The two registers within the SPC contain respectively the algorithm and architecture parameters. In our case the algorithm parameters are related to the type of algorithm (i.e. AES), to the execution mode of the primitive (i.e. feedback, non-feedback) and to the key and data sizes. The architecture parameters are focusing on the reliability (i.e. no, fault detection, fault tolerance), on the throughput, the area (use rate of the device) and the energy consumption.

The next section provides a comparison between several implementations of the AES datapath to define the performance and the costs of security. Then, the last section discusses the efficiency of the whole AES security primitive.

A. *AES datapath implementations comparison*

Four different datapaths have been considered to demonstrate the flexibility provided within the primitive, feedback mode (FB), non-feedback mode (N_FB), feedback mode with fault detection (FB_FD), and feedback mode with fault tolerance (FB_FT). For that study we have considered a 128-bit key. Feedback solutions provide throughput on average hundreds of Mbits/s whereas the non-feedback solution is around the Gbits/s. Fault detection mechanisms enable the system to detect if a fault occurs during the computation of the AES algorithm but without correcting the result. In our case we have used a parity-based technique to detect the fault [19]. Fault tolerance mechanisms provide a tamper resistant architecture. We have considered a TMR technique as it provides an efficient solution [20].

For that study, we have used the Xilinx ISE Foundation 6.3i tool to implement the different datapaths. Power estimations have been performed using the Xilinx XPower 6.3i tool. As shown in table I, each solution corresponds to different levels of performance in terms of area, throughput and power. The highest throughput (3151.1 Mbits/s) is obtained with the non-feedback mode as all rounds are computed in parallel and pipelined but the area and the energy for this mode are also the highest ones (respectively 13689 slices and 1724 mW). Depending on the state of the system, lower throughput or higher reliability have to be considered. Fault tolerance solution is the most secure one but the area and energy overheads are very high (respectively 6302 slices and 1673 mW). Fault detection using parity code does not lead to a significant difference in area and power consumption, respectively +2.1% of slices and -2.7% of power consumption compared to a non secured implementation of AES in feedback mode. For the three feedback mode implementations the throughput is almost equivalent.

Another metric is interesting to compare these implementations, the energy efficiency which represents the throughput per energy (Gbits/J). The non-feedback solution is the most efficient solution but its main drawback is related to its lack of reliability, a fault injection could compromise the whole system. Feedback with and without fault detection provide the same efficiency. Fault tolerance guarantees the security of the primitive but has a high overhead in energy efficiency. Thus, fault detection is a good compromise to guarantee the performance and to increase the security of the primitive and could be considered as an implementation by default.

Figure 2 allows the comparison of the energy efficiency between FPGA, ASIC and processor. ASIC implementation is the best in term of performance (2 decades better in term of Gbits/J compared to the fault detection implementation). But ASIC does not provide any flexibility (*flexibility gap*). The implementation has to be always secured or never, there is no tradeoff. Processor implementations provide flexibility but their performances are poor compared to an FPGA implementation (*processing gap*). Hence, FPGA implementation is the best compromise in terms of performance and flexibility. Furthermore, depending on the security policy several implementations can be considered and dynamically reconfigured.

B. *AES reconfigurable security primitive efficiency*

The three previous feedback implementations (FB, FB_FD, FB_FT) have been considered for the definition of the whole AES security primitive. The non-feedback solution consumes to many slices (99% of our device) to manage its integration within our architecture and to allow dynamic reconfiguration. We have defined three reconfigurable modules which are the datapath, the SPC and the SSC. An area constraint has been associated to each module as shown in figure 3. In this experiment we have considered a single primitive but there is no limitation concerning that point. The communications between the modules has been performed through 3 bus macro which are pre-defined Xilinx hard IPs. One bus macro is used to provide the *fault* signal between the datapath and the SSC. The two others are used between the datapath and the SPC and correspond to control signals (e.g. *start*, *reset*, *done*). The reconfiguration is performed by the SPC through the ICAP interface which allows the dynamic and partial self-reconfiguration of the FPGA. The reconfiguration is dynamic since it is performed while the system is running which is mandatory to react to an attack or to adapt the computation based on the performance policy. It is partial since only the datapath is changed and it corresponds to self-reconfiguration

AES version	Slices		Period (ns)	Frequency (MHz)	Power		Energy (nJ)	Throughput		Energy efficiency (Gbits/J)
	(% of the total amount)	(% compared to FB)			(mW)	(% compared to FB)		(Mbits/s)	(% compared to FB)	
FB	2192 (16%)	-	26.4	37.8	996	-	316	403.7	-	0.4
FB_FD	2240 (16%)	+2.1	25.3	39.4	970	-2.7	295	420.9	+4	0.4
FB_FT	6302 (46%)	+65.2	25.2	39.6	1673	+40.5	507	422.2	+4.4	0.25
N_FB	13689 (99%)	+83.9	40.6	24.6	1724	+42.2	70	3151.1	+87.7	1.8

TABLE I

PERFORMANCE COMPARISON OF THE FOUR AES CONFIGURATION (I.E. DATAPATH). EACH CONFIGURATION CORRESPONDS TO A SPECIFIC TRADEOFF BETWEEN THE SECURITY LEVEL AND THE PERFORMANCE

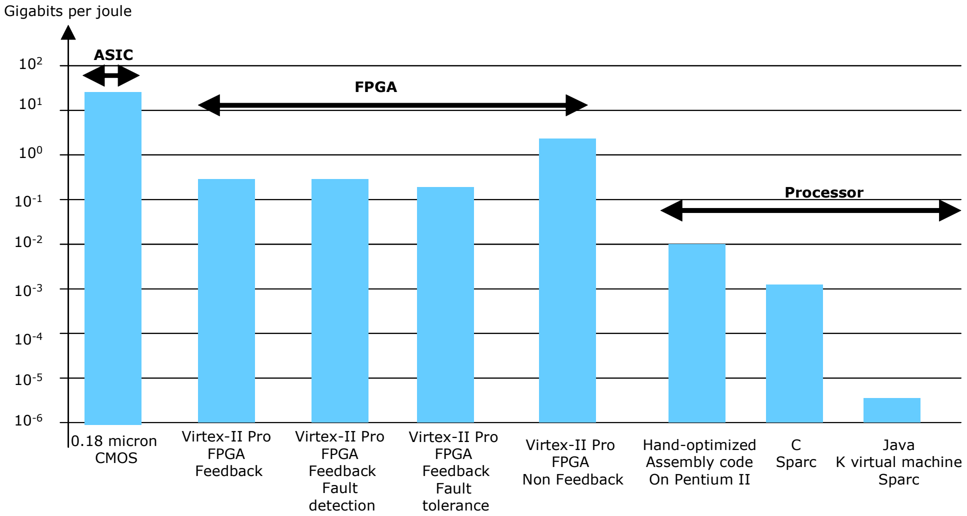
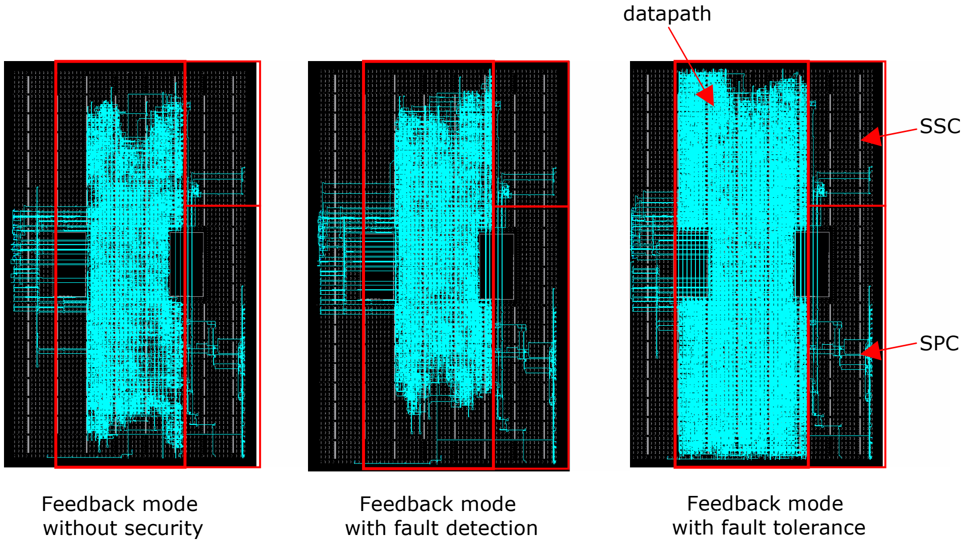


Fig. 2. Energy efficiency comparison of ASIC, FPGA and processor implementations of the AES algorithm. ASIC and processor figures are obtained from [13]



Feedback mode without security

Feedback mode with fault detection

Feedback mode with fault tolerance

Fig. 3. Layout of the three configurations of the AES reconfigurable security primitive. Three modules are defined which are the datapath, the SPC and the SSC

since the SPC reconfigures its associated datapath. Figure 3 shows the three possible configurations. As we can see, the area overhead for the fault tolerant implementation is high compared to the two other solutions. The SPC and SSC modules are very small and remain constant for the three configurations. Their complexity is small compared to the datapath so that they represent a negligible area overhead. For this study we have considered very simple performance and security policies which are basically based on a threshold crossing or on an attack or a fault detection. For real embedded systems, these policies might use more advanced techniques. However, the overhead costs should remain small compared to the datapath.

Concerning the performance of such a solution, the reconfiguration time is directly related to the size of the bitstream. The full bitstream which is used at power up represents 1415 kB and the three partial bitstreams for the FB, FB_FD, FB_FT configurations are respectively equal to 356 kB, 356 kB and 463 kB. In our case the clock of the ICAP interface is 50 MHz which leads to an average reconfiguration time around 8 ms. Each time a reconfiguration is performed there is also an overhead cost in term of power. However, this overhead is negligible for the FPGA power core and represents an increase of around 6% for the FPGA power supply [21].

The results presented in both previous sections show the benefit of using reconfigurable technologies to increase the reliability, the performance and the flexibility of security primitives. Dynamic reconfiguration allows the system to adapt its behavior depending on its state and on the environment state. Such a solution is very promising to provide more secure embedded systems.

V. Conclusion

We have presented a reconfigurable security primitive that leverages the security of embedded systems. The main concepts that drive the definition of this architecture is to use reconfigurable hardware to provide various levels of protection and performances. Such a solution mitigates the design challenges for security and is to our knowledge an original work that enables targeting both security standards and attacks on the system. Results on the AES algorithm demonstrate that the flexibility of our solution enables the definition of an energy-efficient solution while guaranteeing the security. Future work includes the precise definition of sensors to detect attacks. This point is important as low complexity solutions have to be defined not to increase prohibitively the global cost of the system. Solutions based on signatures seem promising and will be investigated. Performance and security policies are also major issues and will be addressed within this research project.

VI. Acknowledgment

This work is supported by the French DGA DSP/SREA under contract no. ERE 04 60 00 010

References

[1] S. Ravi, A. Raghunathan, P. Kocher, and S. Hattangady, *"Security in Embedded Systems: Design Challenges"*, ACM Transactions on Embedded Computing Systems, Vol. 3, No. 3, August 2004, Pages 461-491

[2] S. Ravi, A. Raghunathan, and S. Chakradhar, *"Tamper Resistance Mechnisms for Secure Embedded Systems"*, IEEE International Conference on VLSI Design, January 2004

[3] D. Dagon, T. Martin, and T. Staner, *"Mobile Phones as Computing Devices: The Viruses are Coming!"*, IEEE Pervasive Computing, October-December 2004

[4] T. Martin, M. Hsiao, D. Ha, and J. Krishnaswami, *"Denial-of-Service Attacks on Battery-powered Mobile Computers"*, Proceedings of the 2nd IEEE Pervasive Computing Conference, Orlando, Florida, March 2004, pp. 309-318.

[5] ePaynews - eCommerce Statistics,
http://www.epaynews.com/statistics/index.html

[6] P. Kocher, R. Lee, G. McGraw, A. Raghunathan, and S. Ravi, *"Security as a New Dimension in Embedded System Design"*, ACM/IEEE Design Automation Conference, June 2004

[7] A. Elbirt, W. Yip, B. Chetwynd, and C. Paar, *"An FPGA-based performance evaluation of the AES block cipher candidate algorithm finalists"* IEEE Transactions on Very Large Scale Integration (VLSI) Systems, volume 9, issue 4 (August 2001), pp. 545-557

[8] T. Wollinger and C. Paar, *"Security aspects of FPGAs in cryptographic applications"*, Chapter in "New Algorithms, Architectures, and Applications for Reconfigurable Computing", editors Wolfgang Rosenstiel and Patrick Lysaght, Kluwer, 2004

[9] A. Dandalis and V.K. Prasanna, "An Adaptive Cryptography Engine for Internet Protocol Security Architectures" ACM Transactions on Design Automation of Electronic Systems (TODAES), Vol. 9, N 3, July 2004, Pages 333-353

[10] Gogniat G., Burleson W., and Bossuet L., "Configurable computing for high-security/high-performance ambient systems" accepted for the Embedded Computer Systems: Architectures, MOdeling, and Simulation Conference, Samos, Greece, July 18-20, 2005

[11] A. Hodjat and I. Verbauwhede, *"High-Throughput Programmable Cryptocoprocessor"*, IEEE Micro, May-june 2004, pp. 34-45

[12] D. Oliva, R. Buchty, and N. Heintze, *"AES and the Cryptonite Crypt Processor"*, In proceedings of CASES 2003, Oct-Nov 2003, San Jos, California USA

[13] P. Schaumont and I. Verbauwhede, *"Domain-Specific Codesign for Embedded Security"*, IEEE Computer, April 2003

[14] J. M. Wing, *"Beyond the Horizon: A Call to Arms"*, IEEE Security and Privacy. November/December 2003, pp. 62-67.

[15] H.Van Antwerpen, N.Dutt, R.Gupta, S.Mohapatra, C.Pereira, N.Venkatasubramanian, and R.Von Vignau, *"Energy-aware system design for wireless multimedia"* IEEE DATE, Paris, France, Feb. 2004.

[16] J.L.Wong, G.Qu, and M.Potkonjak, *An on-line approach for power minimization in qos sensitive systems"* in IEEE ASP-DAC, 2003.

[17] J. Daemen and V. Rijmen, *"The Design of Rijndael AES-The Advanced Encryption Standard"* Springer-Verlag 2002

[18] www.xilinx.com

[19] K. Wu, R. Karri, G. Kuznetsov, and M. Goessel, *"Parity Based Concurrent Error Detection for the Advanced Encryption Standard"*, International Test Conference 2004 (ITC), 2004, Charlotte

[20] C. Carmichael, *"Triple Module Redundancy Design Techniques for Virtex FPGAs"* Xilinx Application Note 197 (XAPP197) November 1, 2001

[21] J. Becker, M. Huebner, and M. Ullmann, *"Power Estimation and Power Mesurement of Xilinx Virtex FPGAs: Trade-offs and Limitations*, IEEE Symposium on Intregated Circuits and System Design, September 2003