IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

An Area-Efficient Universal Cryptography Processor for Smart Cards

[1]Priyansh Rai, [2]Lagan Sharma, [3]Divyanshu Gupta

[1]Student, [2]Student, [3]Student
[1]Department of Electronics and Communication Engineering,
[1]ABES Engineering College, Ghaziabad, India

Abstract: Cryptography circuits in smart cards and portable electronic devices are crucial for user authentication and secure data communication. These circuits need to be compact, energy-efficient, capable of handling multiple cryptographic algorithms, and deliver good performance. This paper introduces, for the first time, a hardware implementation of three standard cryptography algorithms on a universal architecture. The micro coded cryptography processor is designed for smart card applications, supporting both private key and public key algorithms while meeting the required power and performance standards. Impressively, it is as small as 2.25 mm² using 0.18-µm 6LM CMOS technology. A new algorithm can be implemented simply by updating the contents of the memory blocks, which are made with ferroelectric RAM (FeRAM). FeRAM allows for nonvolatile storage of configuration bits, which only need to be changed when a new algorithm is introduced.

Index Terms **– Cryptography, Computer Security, Microprocessors, Smart Cards.**

I.INTRODUCTION

The rapid expansion of portable electronic devices with limited power and space has created numerous opportunities and challenges for low-power, compact circuit design in VLSI. Devices like cell phones, PDAs, and smart cards have become essential parts of daily life. This widespread use demands careful consideration for their security systems. Unlike computer network security, which can afford larger area and power consumption but focuses on high data throughput (several Gigabits per second), portable devices require security hardware that prioritizes minimal area and power consumption with less emphasis on throughput (several hundred Kilobits per second to a few Megabits per second). These differing needs necessitate a unique approach to designing security systems for portable electronics. Next-generation, multipurpose smart cards, intended for a variety of applications, must incorporate both private (symmetric) and public (asymmetric) key algorithms to meet diverse security requirements.

Private key algorithms with high throughput are ideal for data communication, while public key algorithms with lower throughput are better suited for private key exchange and authentication. For this application, the data encryption standard (DES), advanced encryption standard (AES), and elliptic curve cryptography (ECC) have been chosen, as they are recognized by standards organizations. DES is included for compatibility with older systems, while AES is selected for its high security and throughput. ECC is favored for public key encryption due to its efficiency. The RSA algorithm, though a standard public key method, is not used here for several reasons. First, a 160-bit ECC offers the same security level as a 1024-bit RSA, making ECC more space-efficient. Second, RSA's reliance on binary addition of large numbers requires either slow carry propagation or large, complex adders. Third, the greater bit length of RSA necessitates wider buses, increasing both area and power consumption—both of which are limited resources in smart cards.

Cryptography systems can be implemented in software or hardware. Software implementations support multiple algorithms on the same hardware but are generally slower and may not meet performance specifications. Additionally, they are more susceptible to side-channel attacks, which exploit physical measurements, like power consumption, to uncover encryption keys. In contrast, hardware implementations that support high throughput lack flexibility, making them unsuitable for smart cards. Similarly, field-programmable gate arrays (FPGAs) are not ideal due to their large area and power requirements, which are not feasible for smart cards.

	Algorithm	HW/ SW	ASIC/FPGA	Clock (MHz)	Area/Gate count	Throughput	Power	Technology
[15] 2000	DES RSA Blowfish SAFER	HW	ASIC	77	-	44.0 MBytes/s 300 Kbps	No Silicon	2µm
[30] 2001	ECC	SW	-	-	-	44.8ms/PM (160bits)	-	ARM (32 bit)
[10] 2001	RSA ECC	HW	ASIC	50	2.9mm x 2.9mm ($8.41mm^2$)	8.2ms (512bit) 6.95ms/PM (176bit)	75mW @2V	0.25µm
[31] 2001	AES	HW	ASIC	333	0.1 mm^2	24Mbps	No Silicon	0.18µm
[32] 2002	ECC	SW	-	13	-	~22ms/PM (163bits)	-	ARM (32bit)
[11] 2002	ECC	HW	FPGA (Variable n)	30	936 CLBs (155bit)	10.8 ms (155bit)	-	Xilinx XCV1000-6
[12] 2002	ECC	HW	ASIC	13.56	14434 gates (160bit) ($0.5mm^2$)*	15ms/PM (160bit)	No Silicon	0.35µm
[14] 2003	AES	HW	ASIC	154	173000 gates ($5.9mm^2$)*	1.6Gbps (AES-128)	56mW @1.8V	0.18µm
[29] 2003	AES	HW	ASIC	465	28626 gates (1.0 mm^2)* + 4Kbit RAM + 128Kbit ROM	1.64Gb/s	314mW @1.8V	0.18µm
Our work	DES AES ECC	HW	ASIC	13.56	2.25mm^2	3.5Mbps 1.83(Enc), 0.85(Dec)Mbps 417ms/PM (155b)	15.9mW 16.3mW 18.3mW @1.8V	0.18µm 6LM

Fig 1.1: Some recent Cryptographic Algorithms Implementation Specifications

According to the International Standard Organization (ISO) standards for smart cards, the maximum allowable chip area is 25 mm². Given the need to include nonvolatile and volatile memories, a CPU, and other peripheral circuits, it's crucial to minimize the area occupied by the security subsystem. This study aims to determine the smallest possible chip area needed to implement security circuits that meet requirements for algorithm agility, power consumption, and throughput. In Fig 1.1, specifically rows 3, 8, and 9, illustrates the area and power consumption of recently published ASIC implementations of various cryptographic algorithms.
Although these implementations mostly target high throughput, they highlight the difficulty of fitting three separate dedicated circuits (for DES, AES, and ECC) into a small area. Additionally, the cryptographic circuits must handle encryption/decryption at the maximum data transfer rate of 847.5 kbps with a clock frequency of 13.56 MHz for contactless smart cards. Since contactless smart cards receive power via RF signals, the available power is limited to 10–20 mW, making power-efficient cryptographic circuits essential. While RISC processors running cryptographic algorithms in software offer attractive solutions for throughput, area, and power consumption, concerns about their vulnerability to side-channel attacks persist. Thus, designing hardware circuits that meet the specifications and resist such attacks is crucial.

Given these constraints, creating an algorithm-agile, area-efficient, and power-efficient crypto processor for smart cards that meets performance and security requirements is challenging. The goal is to achieve this in a design that occupies just 2.25 mm² in 0.18-µm 6LM CMOS. Current research often addresses only parts of this problem. For instance, some works focus solely on energy-efficient reconfigurable cryptography processors for public key cryptography, while others deal exclusively with ECC or DES or AES implementations. There are also FPGA implementations that lack support for both AES and ECC. Additionally, some designs, such as those reconfiguring architectures for six AES competition candidate algorithms, address specific needs but not the full range of requirements.

II.THEORATICAL DISCUSSION

2.1. Data Encryption Standard (DES)

The Data Encryption Standard (DES) is a long-standing algorithm, established in 1977, and has been widely used in military and commercial data exchange and storage for over two decades. DES encrypts and decrypts data blocks of 64 bits using a 56-bit key. The encryption process begins with an initial permutation (IP) of the data block, followed by 16 rounds of a complex, key-dependent permutation, and concludes with a final permutation, which is the inverse of the IP. This process is illustrated in Fig. 2.1. The core function of this algorithm, denoted as $f()$ $f()$, consists of an expansion, XOR operation, lookup table (LUT), and permutation. To decrypt a message, the same algorithm is applied to the encrypted data block using the same key.

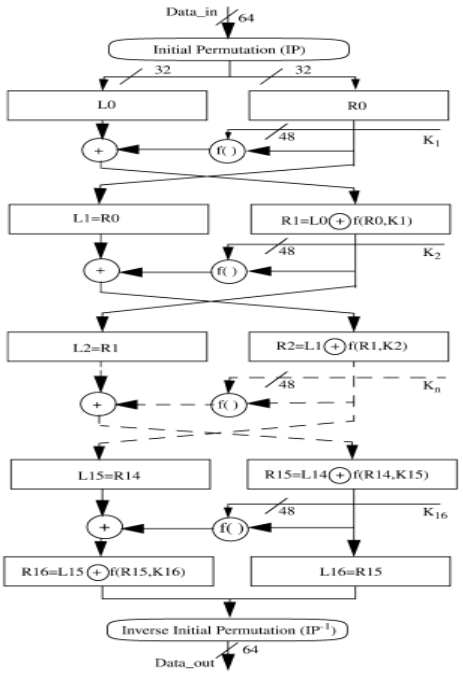

Fig 2.1: DES Block Diagram

Due to the significant increase in computing power since DES was introduced, a brute-force attack—trying all possible key combinations to decrypt an encrypted message—can now be performed relatively quickly, potentially within minutes. Because of this vulnerability, the National Institute of Standards and Technology (NIST) no longer considers DES secure for many applications. A more secure variation endorsed by NIST is the Triple Data Encryption Algorithm (Triple DES, 3DES, or TDEA), depicted in Fig. 2.2. In this variation, DES represents encryption and $DES{-}1DES{-}1$ represents decryption. Triple DES involves applying DES, then, followed by another DES, using three different key options. This process significantly increases the difficulty of breaking the encryption.

The implementation of DES requires only four basic operations: XOR, shift, lookup table (LUT), and permutation. These operations are relatively simple to implement in hardware. The Triple Data Encryption Algorithm (TDEA) also utilizes the same set of operations as DES.

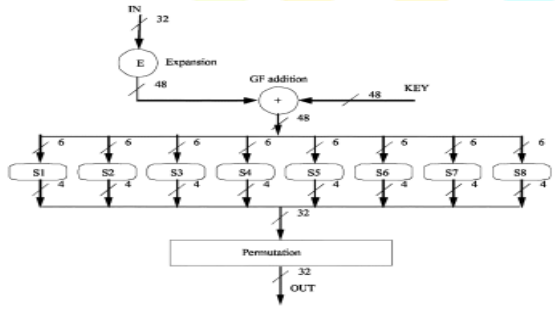

Fig. 2. DES f()-box details.

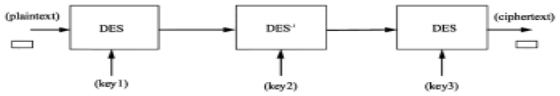

Fig 2.2: 3DES Block Diagram

2.2. Advanced Encryption Standard (AES)

III.IMPLEMENTATION DETAILS

3.1. Encryption Standards

This section details the implementation techniques used in the hardware and/or micro codes of the DES, AES, and ECC algorithms.

3.1.1. DES

The main flow of DES, as shown in Fig. 2.1, involves parameters that are 32 bits wide, making the implementation of the operations in each round straightforward. However, the operations within the \(f() \)-box (illustrated in Fig. 2.1) are slightly modified to fit the architecture. Additionally, the key scheduler is adjusted to ensure its output aligns with the changes made in the \(f() \)-box.

3.1.2. AES

Fig. 2.3 introduced the basic operations of AES. Here, we focus on the details of three fundamental operations (and their inverses) that are crucial for an area-efficient implementation:

1. Sub Bytes: This operation substitutes each byte in the state matrix with another byte using a substitution table (S-box). Its inverse, Inv Sub Bytes, reverses this substitution.

2. Shift Rows: This operation shifts the rows of the state matrix by different offsets. Its inverse, Inv-Shift Rows, shifts them back to their original positions.

3. Mix Columns: This operation mixes the columns of the state matrix by multiplying them with a fixed polynomial. The inverse operation, Inv Mix Columns, reverses this mixing.

The careful implementation of these operations ensures that the overall area requirement is minimized without compromising the algorithm's efficiency and security.

3.2. Cryptography Methodologies*:*

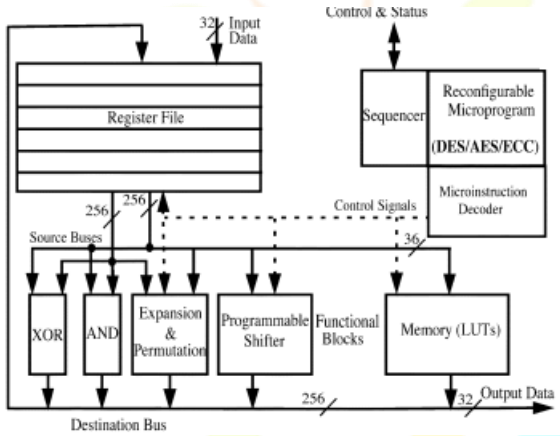

Fig 3.1: Cryptography Processor Architecture

3.2.1. Add Round Key and Inv-Add Round Key:

For AES encryption (decryption), key generation can occur either in one step or in every round. The former approach generates all required key bits for all rounds in advance, requiring extra space to store them until they are used. While this method allows for reusing generated bits for consecutive blocks, our design opts for the latter approach to minimize area usage. In this approach, key bits for each round are produced as needed, eliminating the need for extra space but requiring reproducing key bits from the input key for each block. Although this approach slightly reduces throughput, the impact is more significant for decryption than for encryption. However, the throughput still meets the specifications for this application. Once the key bits for a round are ready, the Add Round Key (Inv Add Round Key) operation is executed on all columns of the state matrix per clock using the XOR operation.

3.2.2 Shift Row and Inv-Shift Row:

Both Shift Row and Inv-Shift Row operations are performed using the Byte Move (BMOV) microinstruction, allowing byte transfers among registers in the register file.

3.2.3 Mix Column and Inv-Mix Column:

These operations are the most complex in the AES algorithm. Their direct implementation requires an 8-bit Galois field multiplier. However, by carefully selecting parameters, these operations can be implemented using only shift and XOR operations.

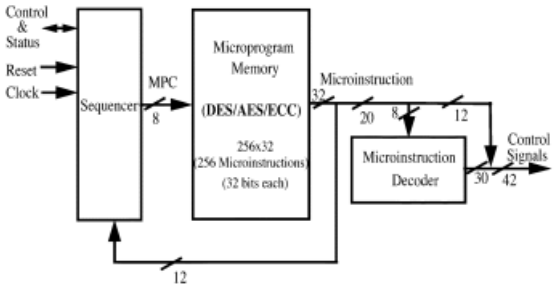

Fig 3.2: Crypto-processor Controller

CRYPTOPROCESSOR CORE POWER CONSUMPTION ESTIMATES PER ALGORITHM (POST-LAYOUT @VDD = 1.8 V)

	Core power Consumption (mW) @fCLK=13.56MHz	Core power Consumption (mW) for 847.5 Kbps throughput @fCLK<13.56MHz
DES	15.9	3.85
AES	16.3	7.55 (Encryption), 16.3 (Decryption)
ECC	18.3	not applicable

Fig 3.3: Crypto Processor Core Power Consumption Estimates

3.3. Area and Power Requirements

We implemented the crypto processor using the Verilog Hardware Description Language and synthesized the RTL code using Synopsys with the TSMC 0.18-µm CMOS standard cell library and SRAM memory blocks. The design comprises five memory blocks (LUTs), 27,874 standard cells, and 18,350 nets. Cadence First Encounter was employed to perform the layout placement and routing, with a core area of 2.25 mm² (1.5 mm x 1.5 mm), accounting for 9% of the total available chip area.

The estimated core power consumption for different algorithms is presented in Fig 3.3, assuming a 1.8-V power supply and a clock frequency of 13.56 MHz. The last column in the table indicates the core power consumption at a reduced clock frequency (below 13.56 MHz) necessary to achieve the required throughput of 847.5 kb/s for contactless smart cards. Power consumption estimation was conducted using the Synopsys Power Analysis tool, utilizing the post-layout netlist of the crypto processor along with node activity data for each algorithm. Further reductions in power consumption can be achieved by operating the processor at voltages lower than the nominal 1.8 V.

THROUGHPUT ESTIMATES FOR THE CRYPTOPROCESSOR OPERATING AT f_{CLK} = 13.56 MHz

	Encryption	Decryption
DES	3.50Mb/s	3.50Mb/s
AES	1.83Mb/s	0.85Mb/s
ECC-83	9.92 PM/s	
ECC-83	3.97PM/s	
ECC-83	2.36PM/s	
ECC-83	0.85PM/s	
ECC-83	0.70PM/s	

(PM/s: No. of Point Multiplications per Second)

Fig 3.4: Throughput Estimates for the Crypto processor operating at f_{CLK} = 13.56 MHz

3.4. Design Performance

We conducted simulations using Verilog-XL for both the RTL and gate-level netlists of the designed crypto processor. The performance of the post-layout design is summarized in Table IX for all three algorithms for ECC specifically.

The first two rows in Fig 3.4 indicate that the throughput of encryption and decryption for DES, as well as encryption for AES, significantly exceeds the maximum data transfer rate for contactless smart cards, which is 847.5 kb/s. Consequently, for applications with stricter power requirements, lowering the clock frequency for these cases can effectively reduce power consumption.

3.5. Output of the Verilog Code

The output of the Verilog Code is shown in Fig 3.5.

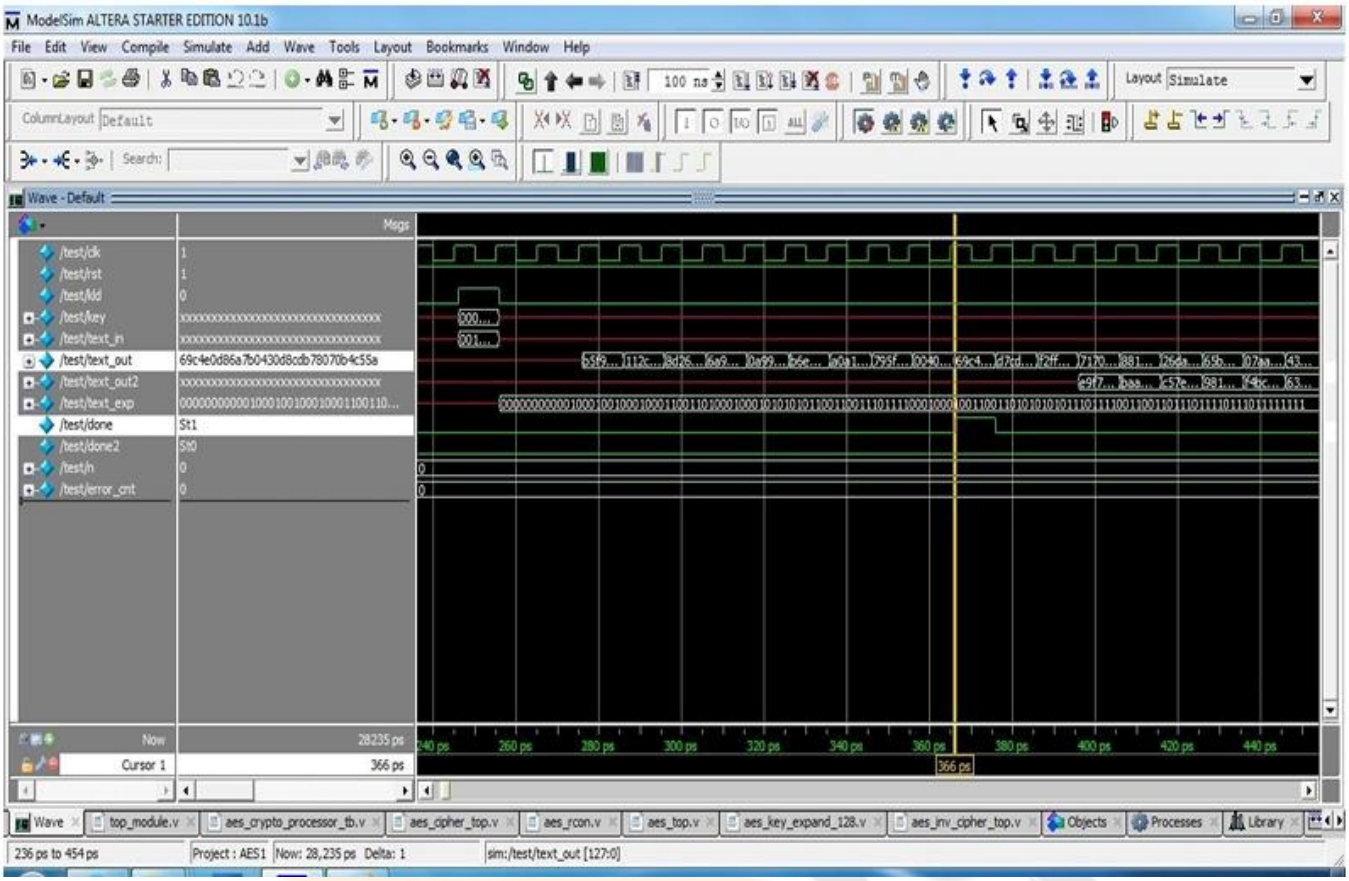

Fig 3.5: Output of the Verilog Code

IV. FUTURE SCOPE

In this section, we explore several additional features that could be integrated into the design to enhance its capabilities:

1. **Microprogram Memory Optimization:** Currently, the design utilizes a 256 x 32 microprogram memory, whereas the maximum number of microcode words needed for DES, AES, and ECC are 46, 150, and 60, respectively. Thus, in the current setup, the microcode for all three algorithms can be stored in the microprogram memory, and algorithm instantiation is achieved by modifying the LUT contents. However, in more space-constrained applications, the memory size could be reduced to 150 words, resulting in a % reduction in area. In this scenario, both the LUT contents and microprogram memory contents would need to be updated by the host CPU during an algorithm change.

2. **ECC Multiplication Hardware Acceleration:** The ECC multiplication is currently executed in a firmware loop, which can degrade performance due to the regular fetching of loop microcode. Implementing this instruction in hardware can potentially improve ECC performance by 200%–300% with minimal area overhead. Additionally, this optimization could reduce power consumption by decreasing the memory access rate.

3. **AES Decryption Optimization:** The current implementation of AES decryption is slower than AES encryption, primarily because of the inverse-key-generation algorithm. Significant improvements could be achieved by reevaluating the inverse-key-generation algorithm and making the registers in the register file byte-addressable.

4. **Support for Additional Algorithms:** While the crypto processor is designed to support DES, AES, and ECC, it offers basic micro-operations that are common in other cryptography algorithms. Therefore, algorithms like Serpent (also a NIST finalist during AES selection) that utilize the same basic micro-operations could be implemented on the same hardware with appropriate microcode development.

5. **Power Consumption Optimization:** While the current design prioritizes feasibility, area requirements, and performance—critical for all types of smart cards—less emphasis has been placed on power consumption, which is a specific constraint for contactless smart cards. Future iterations could focus on optimizing power consumption to better suit the needs of contactless smart card applications.

To further reduce power consumption, designers can employ the following techniques:

1. **Hierarchical Implementation:** By structuring the design hierarchically instead of using a flat implementation, bus wire lengths can be minimized. This reduction in wire lengths decreases both capacitance and switching activity, leading to lower power consumption. Hierarchical implementation allows for better organization of modules and reduces signal propagation delays.

2. **Selective Clocking:** Another method to reduce power consumption is by using a selective clock signal for registers. In cases where operands occupy less than the full register width, only the active portion of the registers needs to be clocked. By selectively activating clock signals for specific register portions, unnecessary switching activity can be avoided, resulting in power savings without compromising functionality.

V. CONCLUSION

In conclusion, this design introduces a groundbreaking universal cryptography processor tailored for smart-card applications, accommodating both private and public key cryptography algorithms. Our achievement lies in expressing the primitives of three pivotal algorithms for smart cards—DES, AES, and ECC—in terms of simple logical operations, maximizing the commonality among them. This innovative approach has yielded a crypto processor that not only meets the power consumption and performance specifications of smart cards but also occupies a mere 2.25 mm² in 0.18-µm CMOS when utilizing SRAM memory blocks. Remarkably, this area accounts for just 9% of the maximum available smartcard die area of 25 mm². Moreover, employing FeRAM instead of SRAM memory blocks offers nonvolatile configuration without any additional area overhead. This design sets a new standard for efficiency and versatility in smart-card cryptography processors, paving the way for enhanced security and functionality in future smart-card applications.

VI.REFRENCES

[1] Data Encryption Standard (DES), Oct. 1999. Fed. Inf. Process. Standards Pub.
[2] Advanced Encryption Standard (AES), Nov. 2001. Fed. Inf. Process. Standards Pub.
[3] IEEE Standard Specifications for Public-Key Cryptography, Jan. 2000.
[4] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, “Investigation of power analysis attacks on smartcards,” in Proc. USENIX Workshop Smartcards Technology, Chicago, IL, May 1999.
[5] K. Okeya and K. Sakurai, “A multiple power analysis breaks the advanced version of the randomized addition-subtraction chains countermeasure against side channel attacks,” in Proc. IEEE Inf. Theory Workshop, 2003.
[6] S. B. Ors, F. Gurkaynak, E. Oswald, and B. Preneel, “Power-analysis attack on an ASIC AES implementation,” in Proc. Inf. Technol.: Coding Computing, vol. 2, 2004.
[7] Smart Cards Standards, 1995–2004. Int. Standard Org.
[8] International Standard Organization/International Electrotechnical Commission ISO/IEC 14 443 standard.
[9] Available: http://www.mips.com/ProductCatalog/P_MIPS324KFamily/product Brief
[10] J. Goodman and A. P. Chandrakasan, “An energy-efficient reconfigurable public-key cryptography processor,” IEEE J. Solid-State Circuits, vol. 36, no. 11, pp. 1808–1820, Nov. 2001.
[11] P. H. W. Leong and I. K. H. Leung, “A micro coded elliptic curve processor using FPGA technology,” IEEE Trans. Very Large-Scale Integer. (VLSI) Syst., vol. 10, no. 5, pp. 550–559, Oct. 2002.
[12] J. H. Kim and D. H. Lee, “A compact finite field processor over GF (2 m) for elliptic curve cryptography,” in Proc. ISCAS, vol. 2, pp. 340–343.
[13] S. Masui, T. Ninomiya, M. Oura, W. Yokozeki, K. Mukaida, and S. Kawashima, “A ferroelectric memory-based secure dynamically programmable gate array,” IEEE J. Solid-State Circuits, vol. 38, no. 5, pp. 715–725, May 2003.
[14] Yadollah Eslami, Ali Shiekholeslami, P. Glenn Gulak, Shoichi Masui, Kenji Mukaida, “An Area-Efficient Cryptography Processor for Smart Cards”, IEEE Transactions on Very Large-Scale Integration (VLSI) Systems, February 2006.