AN AREA-EFFICIENT UNIVERSAL CRYPTOGRAPHY PROCESSOR FOR SMARTCARDS

Mr.G.RamanaKumar
M.Tech in Digital Electronics & Communication
ECE Department,
Chaitanya Engineering College,
Vishakapatnam,(A.P).

Mr.K.Suresh
Head of Department
ECE Department,
Chaitanya Engineering College,
Vishakapatnam,(A.P)

Ms.B.Swati,
Assist. prof in Dept Of E.C.E,
Chaitanya Engineering College,
Vishakapatnam,(A.P)

Abstract

Cryptography circuits for smart cards and portable electronic devices provide user authentication and secure data communication. These circuits should, in general, occupy small chip area, consume low power, handle several cryptography algorithms, and provide acceptable performance. This paper presents, for the first time, a hardware implementation of three standard cryptography algorithms on a universal architecture. The microcoded cryptography processor targets smart card applications and implements both private key and public key algorithms and meets the power and performance specifications and is as small as 2.25 mm2 in 0.18- m 6LM CMOS. A new algorithm is implemented by changing the contents of the memory blocks that are implemented in ferroelectric RAM (FeRAM). Using FeRAM allows nonvolatile storage of the configuration bits, which are changed only when a new algorithm instantiation is required.

Key words: Computer security, cryptography, microprocessors, smart cards.

Introduction

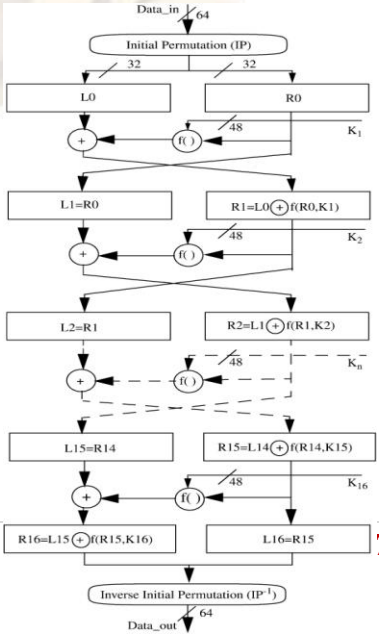
The rapid growth of portable electronic devices with limited power and area has opened a vast area of low-power and compact circuit design opportunities and challenges for VLSI circuit designers. Cellular phones, PDAs, and smart cards are examples of portable electronic products that are becoming an integral part of everyday life. The popularity of these devices necessitates special considerations for their security subsystem. Unlike computer network security systems that impose less stringent limitations on the area and power consumption but put more emphasis on high throughput (several Gigabit/s), portable applications demand security hardware with more restrictions on area and power and less on throughput (several hundred kilobit/s to a few Megabit/s). This difference in requirements dictates a different approach in the design and implementation of the security systems for these devices. Since next-generation, multipurpose smart cards will be used for a wide range of applications, their security system must implement both private (symmetric) and public (asymmetric) key algorithms, to accommodate various application requirements.

Private key algorithms with high throughput are suitable for data communication, while public key algorithms with much lower throughput are suitable for private key exchange and authentication. Among all available algorithms, data encryption standard (DES), advanced encryption standard (AES), and elliptic curve cryptography (ECC), which are approved by standards organizations, are selected for this application.

CRYPTOGRAPHY ALGORITHMS

A. Data Encryption Standard (DES)

This is a well-established algorithm that has been used for more than two decades (since 1977) in military and commercial data exchange and storage. The algorithm is designed to encipher and decipher blocks of data consisting of 64 b using a 56-b key. A block to be enciphered is subjected to an initial permutation (IP), then to 16 rounds of a complex key-dependent permutation, and, Nelly, to another permutation which is the inverse of the IP, IP^{-1}, as shown in Fig.



The function f() in this figure is the heart of this algorithm and consists of an expansion, XOR, lookup table (LUT), and permutation, as depicted in Fig. 2.

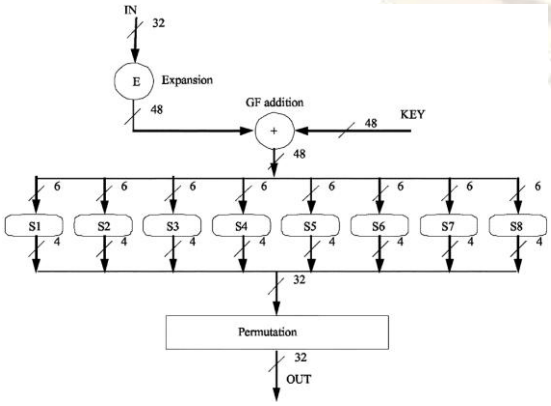
To decipher, it is necessary to apply the very same algorithm to an enciphered message block, using the same key.

Since the processing power of current computers is much higher than those of two decades ago, a brute-force attack (checking all possible key combinations to decipher an encrypted ciphertext) to this algorithm is possible in a relatively short time (possibly as short as a few minutes [21]). For this reason, this algorithm is no longer considered to be a secure algorithm for many applications by the National Institute of Standards and Technology (NIST). A more secure algorithm based on DES which is still supported by NIST is called the triple data encryption algorithm (Triple DES, 3DES, or TDEA) depicted in Fig. 3. In this figure, DES represents encryption and DES^{-1} represents decryption. 3DES involves applying DES, then DES^{-1}, followed by a final DES to the plain text using three different key options , which results in a cipher text that is much harder to break.

The implementation of DES needs four basic operations only, namely, the XOR, shift, LUT, and permutation, which are relatively simple to implement in hardware. The TDEA also uses the same set of operations as DES.

B. Advanced Encryption Standard (AES)

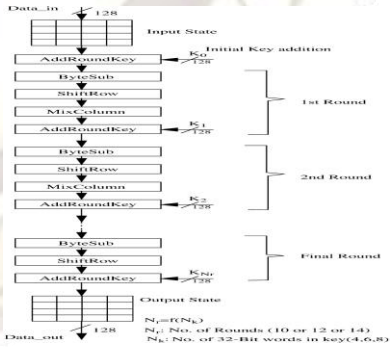
AES, also known as Rijndael, is a block encryption algorithm which encrypts blocks of 128 b using a unique key for both encryption and decryption. A block diagram representation of the algorithm is shown in Fig. 4. Three versions of the algorithm are available differing only in the key generation procedure and in the number of rounds the data is processed for a complete encryption (decryption) . AES-128 uses a 128-b key and needs 10 rounds. AES-192 and AES-256, respectively, need 192-b and 256-b keys and 12 and 14 rounds for processing a block of data.



The 128-b input data is considered as a 4 4 array of 8-b bytes (also called "state" in the algorithm). The state undergoes four different operations in each round, except for the final round which has only three operations. These operations are "*ByteSub*," "*ShiftRow*," "*MixColumn*," and "*AddRoundKey*" operations. "*MixColumn*" is omitted in the final round. Each round of the algorithm needs a 128-b key, which is generated from the input key to the algorithm. The key-scheduler block (not shown in Fig) consists of two sections: the key expansion unit, which expands the input key bits to the maximum number of bits required by the algorithm, and the key selection unit, which selects the required number of bits from the expanded key, for every round. As mentioned before, aside from the key values, all of the steps in all of the rounds are the same except for the last round that *Mix Column* is not present.

Each byte in the state matrix is an element of a Galois Field GF(2^8), and all of the operations can be expressed in terms of the field operations. In simple terms, $2^n$is a set of elements each represented by an -bit string of 0's and 1's and two basic operations: addition and multiplication. These two operations are defined such that the closure, associativity, and other field properties are satisfied.

Due to the specific choices of the parameters of the algorithm, this operation can be expressed as a matrix multiplication, which can be implemented using shift and XOR operations. *AddRoundKey* is just a logical XOR operation.

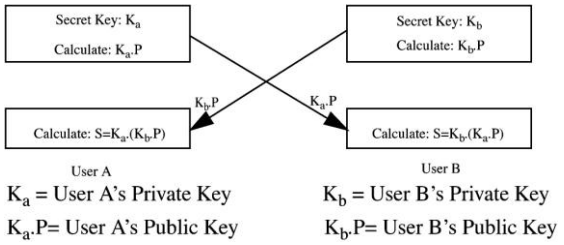


C. Elliptic Curve Cryptography (ECC)

The set of all ( ) pairs satisfying the nonsupersingular elliptic curve equation

$$\mathbf{Y^2+xy=x^3+ a_2 x^2+a_6}$$

are called points on the elliptic curve E, where x, y ,a2 , and $\mathbf{a_6}$ are elements of the GF(2^n) The *point addition (S=P+Q)* and *multiplication* (R = K.p where K is a constant) operations are defined such that both S and R are also points on the elliptic curve E. Moreover, knowing R and P, it is practically impossible to find . This property forms the fundamental foundation of ECC.



which is the secret key exchange. The basic secret-sharing algorithm (ECKAS-DH in [3]), also known as the Diffie–Hellman protocol for key exchange, is pictured in Fig. 5. In brief, both users, A and B, agree on the elliptic curve, a point on and a mathematical basis, such as polynomial basis or normal basis (NB). Each user then chooses a secret key from GF (2^m), K_a and K_b and calculates her/his own public key (PKa = Ka. P and $Pk_b = K_b$. P) and sends it to the other user. At this point both users can calculate the secret point both users can calculate the secret point. Note that, although both and are available, only one of them should be used for higher security.

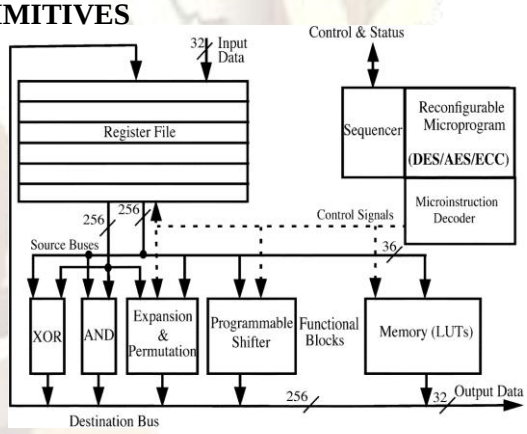
CRYPTOGRAPHY ENGINE

Our procedure for designing this domain-specific processor starts by listing all of the required functional blocks for the selected algorithms followed by the selection of a minimum set of the required blocks that can implement all algorithms. Then, a cost function, which is the area in this case, but could be any other parameter such as power or speed or any combination of several parameters, is defined and calculated for the selected set.
If the cost is larger than the available budget, the complex functional blocks are expressed in terms of simpler blocks by revising the algorithm primitives or by selecting a new mathematical interpretation of the operation. This should be followed by the selection of the minimum set of functional blocks and calculation of the new cost. The procedure should be repeated until the cost criterion is met or the simplest form of the functional blocks is achieved.

The list of operations required by the three algorithms DES, AES, and ECC is shown in Table II. The list is directly derived from the primitives of the algorithms as defined in Section II. Referring to Table I, a design including three independent circuits, each implementing one of these algorithms, needs an area that is much larger than the available area for this application. A reconfigurable architecture at this level is not feasible either, because most of the operations are complex and dedicated to each algorithm, and there are only few operations which are common among the three algorithms.

Our approach to address this problem is to express the high level operations of each algorithm in terms of basic arithmetic and logic operations to maximize the number of common operations among the algorithms and, hence, minimize the overall required chip area. This is challenging for two reasons. First, for DES, the data and the key lengths are fixed at 64 and 56 b, for AES the data length is fixed at 128 b, but the key length could be any of 128, 192, or 256 b, and for ECC both the data and the key lengths are variable (-bits). Second, DES uses a set of simple logical functions, substitution, and permutation operations, and AES adds to this set a more complicated multiplication which can be implemented using shift and XOR operations, while ECC is based on complex mathematical finite field operations in GF(2) and The field elements used in ECC can be represented in either polynomial basis (PB) or normal basis (NB) [3], [17]–[19]. The hardware implementation of the field operations when using PB requires large silicon area. By using NB, squaring (GF SQR in Table II) is achieved by a simple circular shift left operation. Furthermore, multiplication is simplified to a series of LUTs, AND, and XOR operations, and inversion is implied to a series of shift and field multiplications.

OPERATIONS REQUIRED BY DES, AES, AND ECC AFTER REORGANIZATION OF PRIMITIVES



Detailed cryptoprocessor architecture

Table shows the list of all required functional blocks for the desired algorithms after expressing the basic primitives in terms of simple operations. The table reveals that the algorithms may be implemented using much simpler operations that also maximize the number of common blocks among them. Based on the information in Table, we are proposing the crypto processor. This design implements all of the required micro operations for the three algorithms and provides algorithm agility by updating the contents of its memory blocks.

CRYPTOPROCESSOR SPECIFICATIONS

This section includes three subsections. Section IV-A explains the architecture of the designed cryptoprocessor, which is followed by the details of its initialization and interface with the host CPU in Section IV-B. Section IV-C introduces the Microinstruction set and the Microprograms for the implemented algorithms in the cryptoprocessor.

A. Architecture

A microprogrammed control unit (Fig. 7) controls the data path and implements the algorithm agility. The design uses a 32-b bidirectional data bus and a 9-b address bus and a 13-b control bus to communicate to the host CPU. The host CPU uses these signals to write/read the algorithm inputs/outputs to/from the memory-mapped cryptoprocessor registers in addition to the updating the configuration memories (microprogram and LUTs) that define the function of the cryptoprocessor.

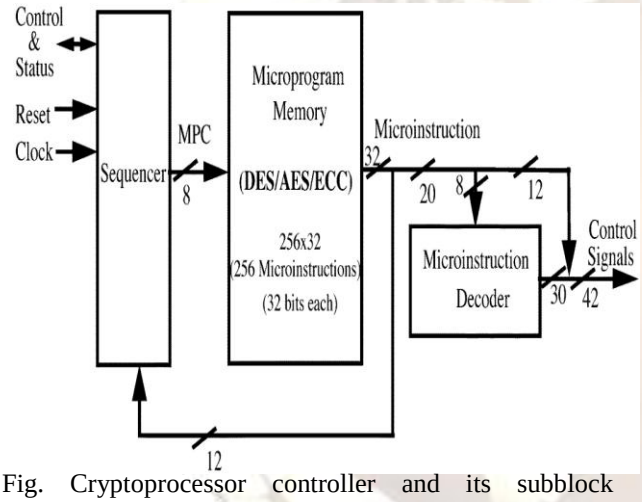


Fig. Cryptoprocessor controller and its subblock interconnections.

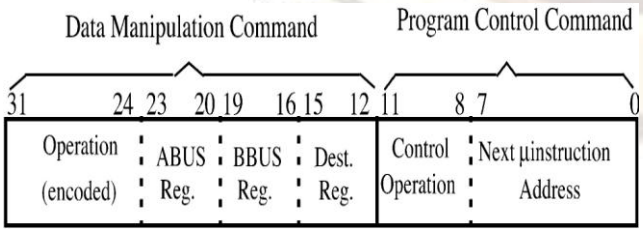


Fig. Cryptoprocessor microinstruction format.

TYPICAL FeRAM AND SRAM CHARACTERISTICS

	Type	Read_mode	Size	Area (mm^2)	Power ($mW/V^2/MHz$)	Access Time (ns)
FeRAM	NonVol.	Dest.	256X32	0.26	0.0213	50
SRAM	Vol.	NonDest.	256X32	0.35	0.0163	1.9

Dest. : Destructive Vol.: Volatile

IMPLEMENTATION DETAILS

This section highlights the implementation techniques used in the hardware and/or the microcodes of DES, AES, and ECC algorithms.

A. DES

The parameters used in the main flow of DES depicted in Fig. are 32 b wide and the implementation of the operations in each round is straightforward [1]. However, the operations required for the f()-box shown in Fig. 2 are slightly modified to match the architecture. Note that the key scheduler is also modified to have its output match the changes in the f()-box.

B. AES

Section II along with Fig. 4 introduced AES operations. This section focuses on the subtleties of three fundamental operations (and their inverses) that are critical to their area-efficient implementation.

NO. OF CLOCK CYCLES REQUIRED PER ALGORITHM.

	Encryption	Decryption
DES	248	248
AES	951	2036
ECC-83	1,367,114	
ECC-83	3,414,850	
ECC-83	5,739,898	
ECC-83	15,945,058	
ECC-83	19,297,332	

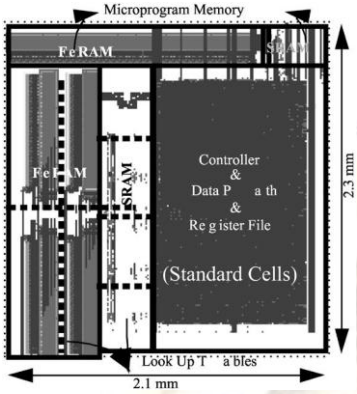
C. ECC

The point multiplication in ECC is the basic operation of this algorithm, and, instead of times the addition of the point to itself, it is more efficient to use the binary ex-pansion of and use a set of doubling and addition to find, as presented in [11]. The point-doubling operation is carried out by the point-doubling microcode and the point addition operation is performed by the point addition microcode. Both of these microcodes use field multiplication and inversion subroutine microcodes. Field addition and doubling is cheap (XOR and Circular shift left) and is done in the point multiplication main routine. It should be emphasized that, since all of the computations are performed in a single basis (ONB), there is no need for the basis conversion, and, hence, no basis conversion overhead is associated with this design.

AREA AND POWER REQUIREMENTS

We used the Verilog Hardware Description Language to realize the crypto processor and Synopsys to synthesize

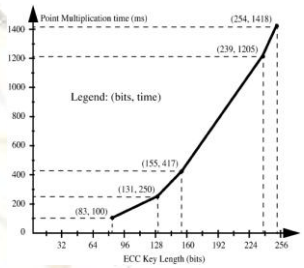
the RTL code using TSMC 0.18- m CMOS standard cell library and SRAM memory blocks. In addition to five memory blocks 27 874 standard cells and 18 350 nets are present in the design. Cadence First Encounter was then used to place and route the layout of the crypto processor, resulting in the layout shown in Fig. 11 with a core area of 2.25 mm (1.5 mm 1.5 mm), which is 9% of total available chip area.



DESIGN PERFORMANCE

We used Verilog-XL to simulate both the RTL and gate-level net lists of the designed cryptoprocessor. The simulated performance of the post-layout design is summarized in Table IX (for all three algorithms) and in Fig. 13 (for ECC only). The first two rows of Table IX show that the throughput of encryption and decryption of DES and encryption of AES algorithms are much higher than the maximum data transfer rate for contact- less smart cards, at 847.5 kb/s [8]; hence, for more power-restricted applications, the clock frequency can be lowered for these cases to reduce the power consumption. Also, note that, for ECC, the performance is measured in point multiplication per second (PM/s) rather than bits per second (b/s), because ECC is mainly used for secret key exchange and authentication purposes which both use point multiplication as the basic operation. The ECC performance plot in Fig. 13 shows that highly secure applications (239 254 b ECC) need to spend around 1.5 s for secret key exchange or a point multiplication, while medium (131, 155 b) and low (83 b) security applications can perform the same operations in less than a second. Moreover, it is possible to trade performance with area and power in this implementation. For example, higher performance can be obtained by running the processor at higher frequencies—up to 25 MHz for the current design—(increasing power consumption) and/or using pipelining (increasing area), for more performance-demanding applications. It is worth mentioning that the performance listed in Table IX and Fig. 13 for ECC are based on the worst case scenario mentioned in Section IV-C.

As mentioned in Section I, there is no published work of a single hardware implementation that supports DES, AES, and ECC available for comparison. Table I lists several recent publications of hardware and software implementations of cryptography algorithms contrasted with the implementation described in this paper. Moreover, the original proposal d**o**cuments of Rijndael to the AES selection committee by the algorithm designers indicates that the fastest implementation of AES-128 on Intel 8051 needs 3168 CPU cycles (each CPU cycle is 12 clock cycles) [33] and 1016 bytes of code, and its implementation on Motorola 68HC08 needs 8390 clock cycles and 919 bytes of memory [33]. Our implementation, however, needs 248 clock cycles and uses 92 bytes of microcode only, which is abut 40 times faster and 10 times more code-size efficient compared to the referenced implementations.



	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)

THROUGHPUT ESTIMATES FOR THE CRYPTOPROCESSOR OPERATING AT f= 13:56 MHz

DISCUSSION

In this section, we address a few extra features that can be added to the design in the future to enhance its capabilities. First, the design currently uses a 256 32 microprogram memory, while the maximum number of microcode words needed by DES, AES, and ECC are 46, 150, and 60, respectively. Thus, in the current design, the microcode for all three algorithms can be stored in the microprogram memory and the algorithm instantiation is accomplished by changing the LUT contents only. In a more area-limited application, the memory size can be reduced to 150 words (%mm reduction). In this case, the microprogram memory contents, in addition to the LUT contents, need to be updated by the host CPU during an algorithm change. Second, the ECC multiplication is currently performed in a firmware loop which degrades the performance due to the regular fetching of the loop microcode. A hardware implementation of this instruction can improve the ECC performance by 200%–

300%, with negligible area overhead. This can also reduce the power consumption by reducing the memory access rate.

Third, the current implementation of AES decryption is much slower than AES encryption, mainly because of the inverse-key generation algorithm. This could be greatly improved by reconsidering the inverse-key-generation algorithm and by making the registers in the register file byte-addressable.

Fourth, although the cryptoprocessor is designed to support DES, AES, and ECC only, it provides basic micro-operations that are used in other cryptography algorithms. Therefore, other algorithms such as Serpent (which was also a NIST finalist during AES selection [28]) that use the same basic micro-operations can also be implemented on the same hardware by developing a suitable microcode for each algorithm. Finally, for the current design, we have been mainly concerned about the feasibility, area requirements, and the performance of the cryptoprocessor, which are important for all types of smart cards, and less concerned about the power consumption, which is a constraint specific to contactless smart cards only. However, we have used several implementation techniques, such as using keeper cells on the bus lines and clock gating for the registers in the register file, to keep the power consumption low. To reduce power consumption further, the following techniques are available to the designers: 1) reducing the bus wire lengths by using a hierarchical implementation, instead of a flat implementation and 2) using a selective clock signal for the registers to clock the active portion of the registers for cases where operands occupy less than the full register width.

CONCLUSION

This design presents, for the first time, a universal cryptography processor for smart-card applications that supports both private and public key cryptography algorithms. We achieved this by expressing the primitives of three important algorithms for smart cards (DES, AES, and ECC) in terms of simple logical operations that maximize the number of common blocks among them. This approach resulted in a cryptoprocessor that meets the power consumption and performance specifications of smart cards and occupies 2.25 mm in 0.18- m CMOS when SRAM memory blocks are used. This area represents just 9% of the maximum available smart-card die area of 25 mm . Using FeRAM instead of SRAM memory blocks provides nonvolatile configuration at no extra area overhead.

REFERENCES

1. Advanced Encryption Standard (AES), Nov. 2001. Fed. Inf. Process Standards Pub.
2. IEEE Standard Specifications for Public-Key Cryptography, Jan. 2000.
3. 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, p. 151 and 161.
4. 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 Work shop, 2003, pp. 175–178.
5. 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, pp. 546–552.
6. Smart Cards Standards, 1995–2004. Int. Standard Org.
7. International Standard Organization/International Electrotechnical Commission ISO/IEC 14 443 standard.
8. [Online]. Available: http://www.mips.com/ProductCatalog/P_MIPS324KFamily/productBrief
9. 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.