Brill Academic Publishers
P.O. Box 9000, 2300 PA Leiden,
The Netherlands

Lecture Series on Computer
and Computational Sciences
Volume 2, 2005, pp. 166-169

Implementation Aspects of Rijndael Encryption Algorithm

M. Shahraki

Department of Electrical Engineering,
University of Sistan & Baluchistan
Email: shahraki@rediffmail.com

Abstract: A cryptographic system is not only needed to protect an stand-alone system, like databases on a PC, operating system, system drivers and programs running on a system, but also it is needed to protect information transmitted between a set of devices, like; router, ATM switches etc [1]. Each one of these applications has unique characteristics and therefore need unique features that cryptography must satisfy. Advanced Encryption Standard (Rijndael) is examined in this paper, concerning software and hardware implementation platforms. This work focuses on the alternative integration approaches of Rijndael suitable for software and hardware applications. It presents the advantages and the trade offs, for alternative approaches, as well as comparison results concerning performance issues. Furthermore this work introduces the design criteria for AES developments in both software and hardware manners.

Keywords: Advanced Encryption Standard, AES, Software, Hardware, Networks, Communications

1. Introduction

Advanced Encryption Standard Ciphers consist of Rijndael, Serpent, Twofish, RC6 and Mars, which Rijndael is the winner of this group and is selected as the future algorithm to protect applications against adversaries. These applications range from wired to wireless [1] systems which all need encryption to transmit their information very secure and fast against adversaries and their attacks. Each one of these applications has unique characteristics and therefore need unique features that cryptography must satisfy. Advanced Encryption Standard (Rijndael) is examined in this paper, concerning software and hardware implementation platforms and their pros and cons.
First of all in section 2, its software implementation, concerning different processor platforms [2, 3, 4] and different soft tools (assembler/compiler) [5, 6] is presented. In section 3, its hardware implementation, concerning different methods, architecture [7] and platforms is noticed. In section 4 other criteria which are important in designing a protected system is considered. We will see that in some applications to satisfy required characteristics, it is needed to combine hardware and software design for a cryptographic purpose.

2. Software Implementation of Rijndael

Cryptography system, in this article, we mean Rijndael, could be implemented as a software program. Such implementation has public advantages, like; ease of use, ease of maintenance, portability and low cost, but it offers disadvantages like; poor security, higher power consumption and lower speed against hardware implementation [1]. Software implementation is influenced with language, compiler, platform, block size, key size, implementation structure and designer's used methodology. It could use C, Java, Assembly and Matlab as a soft tool [2, 3, 5, 6], and also a processor (SISD, SIMD and MIMD) [3], or MicroController (like 8051 [4]), as a hard platform.

Table 1: Clock cycles required for AES

Processor	**Key schedule and Enc. / Dec.** using key unrolling		**Enc. / Dec.** using key on-the-fly	**Implemented in**
ARM7TDMI	634	1675 / 2074	2074 / 2378	[6]
	449	1641 / 2763	1950 / 3221	[5]
ARM9TDMI	499	1384 / 1764	1755 / 1976	[6]
	333	1374 / 2439	1623 / 2796	[5]
ST22	0.22	0.51 / 0.60	0.72 / 0.82	[6]
	0.13	0.61 / 1	0.75 / 1.13	[5]
Pentium III	370	1119 / 1395	-	[6]
	396	1404 / 2152	-	[5]

Table 1, shows the effect of different hard platforms on encryption, decryption and key scheduling algorithms of Rijndael. By optimizing the algorithm it is possible to enhance software speed and memory usage [2], but however this enhancement is not considerable.
Hardware platform can be upon RISC or CISC. RISC has short and simple instructions and regular form in fetching/decoding, pipelining and scheduling, that enables a fast and simple implementation, but it is also fast and simple for adversaries to attack this implementation [8]. Basic parameters that influence software implementation with a special hard platform are: basic underlying architecture, whether it is 32 bits, 8 bits etc, use of different memories, structure of processor (SISD/SIMD/IMD), and its internal operation, number of Cpu ports and the number and length of registers.
As indicated, a soft tool could be Java, C, assembly, Matlab etc. In Java programming (32 bit processor), source code is compiled into byte code instead of machine code [2], so at runtime byte code must be converted to machine code, because Java programs are independent of processor and operating system. Therefore, a sizeable portion of compilation is delayed until runtime that this characteristic slows down Java compared to compilers like; C and also assemblers. Also Java supports 32 bits processors against C and assembly which support 64 bit processors, and this feature influences speed of Rijndael cryptography algorithms execution. On the other hand assembly code has better results (code size and speed) than that the C, because of optimized use of internal architecture of processor and use of complex instructions. Matlab is another alternative [5], because of its best representation for key and blocks of plain/cipher texts.
Anyway key scheduling of Rijndael is about 7 times faster than other AESs, its encryption/decryption is lower than RC6 but not more than 32%. In Java and C implementations, encryption/decryption algorithm of Rijndael is not the fastest, because Rijndael is not c-friendly, but in assembly, Rijndael makes very heavy use of the processors' new technologies. Rijndael is based only on most simple imaginable operations like; load and Xor. On the other hand Rijndael has large internal parallelism ability, and there is a large number of possibilities of reschedule its code, table 2 [6].

Table 2: C implementation of Rijndael compared to other AESs.

Cipher	Mbps on a 450 MHz PII	Clock/block	μ OPs/cycle
Rijndael	243	237	2.54
RC6	258	223	1.47
Twofish	204	282	2.11
Mars	188	306	1.87
Serpent	-	-	-

3. AES Hardware Implementation

Rijndael could be implemented in different modes and architectures, and in a variety of hardware devices like; ASICs, FPGA and CPLD, and also Smart Cards. Hardware platforms work with less frequency than current processors, but it is generally faster than their software equivalent, about 4 times [9], because Rijndael has parallel processing and pipeline characteristics that well suit on hardware platform, of course this is not common for all ciphers. Against to the tight physical security, low power consumption, high execution speed, some cons like difficult implementation, high implementation cost, and etc exist in hardware implementations. These factors are critical items, which must be taken care of special attention of the designers. ASIC designs guarantee better performance, with fast execution, enough small dedicated size and security against software implementation and even FPGA, but it is less feasible regard to both of them. Also FPGA provide faster and easier design, more flexibility and reconfigurability, which is very important for key-agility, as like we see on SSL, TLS etc [1]. FPGA is a middle one that has flexibility of software design and security and speed of hardware design. But even FPGA could be configured at runtime, decide number of pins is necessary before board implementation. Smart card is another option for hardware implementation of Rijndael, but in smart cards the RAM requirements are more important, than the clock frequency. It is cleared that the devices of this category are not proper for large encryption systems with special specifications [1] because of its slow communication with external memories.
Anyway, hardware implementation could be done in feedback (CBC, CFB, and OFB), and non-feedback modes (ECB), and architecture for the encryption/decryption unit could be implemented in one of these methods: Basic iterative architecture, Partial and full loop unrolling, Outer-round pipelining, Resource Sharing. Researches performed in [7] and similar efforts show that in iterative architecture and feedback modes, Serpent and Rijndael have the highest throughput at the expense of the relatively large area. On the other hand Rijndael and Mars use the modest area in feedback mode and iterative architecture [7], as indicated in table 3. Differences between different implementations of

Rijndael is according to different optimization for speed, area and sharing resources, which depends on designer's art, but no many improvement could be achieved.

Table 3: Implementations Comparisons

Cipher	throughput (Mbit/s)	Area (CLB slices)	Throughput/Area ration
RC6	142.7	1137	0.122
Twofish	177.3	1076	0.164
Rijndael	414.2	2507	0.166
Serpent I1	-	-	0.124
Serpent I8	431.4	4507	0.097
Mars	61.0	2744	0.022
3DES	59.1	356	-

For feedback mode, iterative architecture is very suitable, but decision to choose architecture for non-feedback modes is not easy [7], different architecture in non-feedback modes has different effects on Rijndael, table 4 shows affect of different architecture modes in non-feedback mode for Rijndael. In non-feedback mode, throughput is in excess of 3.65 Gbit/s [10], taking into account both FPGA and ASIC implementations. Full mixed-inner-outer round implementation has better throughput than other architecture. Implementation of Rijndael consumes approximately the same amount of FPGA resources like Serpent and Twofish. No correlation between software and hardware performance was found. The difference among Rijndael implementations in software or hardware is based on internal structure of these algorithms.

Table 4: Different hardware implementation of Rijndael

Architecture	Throughput (Mbits/s)
Iterative (feedback mode) [4]	414.2 Mbit/s
Full mixed -inner-outer -round pipelining (non-feedback mode) [4]	12.2 Gbit/s
Full outer-round pipelining (non-feedback mode) [4]	5.7 Gbit/s
Rijndael [10]	3.65 Gbit/s
Rijndael processor [11]	2.29 Gbit/s

4. Other Criteria in Rijndael Implementations

The performance of cryptography in high-speed applications closely requires tradeoff between security and speed, and there are many criteria that must be considered in software/hardware implementation to be able to have better performance and throughput. In some applications speed is more important than other features, and in some of them, power consumption and in a group of applications which have algorithm-independent nature, like; Secure Sockets Layer (SSL), IPsec [1], voice-over-IP products, high-speed routers and ATM switches [1, 7], switching between Rijndael and other ciphers is very important. Rijndael could be implemented to present high speed and low power consumption, e.g. it could present higher speed and lower power consumption and used area for 802.11 protocols, against RC4 [10]. Also Rijndael could be implemented with different hardware methodologies to satisfy low power consumption or high execution speed. In [11] two different VLSI architectures are presented. The first uses feedback logic and reaches throughput value equal to 259 Mbit/sec with low covered area resources and the second is optimized for high-speed performance using pipelining technique with high data throughput of 3.65 Gbit/sec. These two implementations could be used for online cryptographic needs of high speed networking protocols.

But anyway, against all consideration to speed up and decrease covered area, in before sections, there are many attacks from adversaries' side both for software [8, 12] and hardware [8, 13] implementations of not only Rijndael [14] but also all ciphers. Computer hackers often have many techniques, either in hardware and/or software, at their disposal to crack out the secret [8]. Software implementation of cryptography algorithms is based on an operating system, therefore it maybe execute in parallel with other processes that this subject slows it down, and cause to use a common memory (main memory for holding intermediate results and external memory to hold encryption/decryption modules and long term keys). So there is no protection for code, key, and also intermediate results. Power consumption analysis and reverse engineering are other attacks that could disturb software implementations of ciphers and other applications. Also there are some known and special attacks for Rijndael, like; Impossible Differential attack, Square attack and Collision attack [14].

Hardware implementation of Rijndael, does not have software weaknesses which are presented in last

paragraph, but it is natural that similar to software side, there are some attacks against hardware implementations on Smart Cards, ASICs and FPGAs, See [8,13] for more details. Comparing attacks against hardware and software implementation concludes that FPGA and ASICs are currently more secure than its software implementation peers.
Another concept in Rijndael implementation is its co-design. It is very noticeable, because this type of design could be used to improve final product security and speed, especially when key agility between Rijndael and other ciphers is needed.

6. Conclusion

In step with high performance networks and applications, high speed cryptography is needed more than before. It must be considered that high performance applications require an optimal trade off between security and speed. Fast hardware relies on parallelism and pipelining, while in software designs access to memory is a key to gain fast performance. This aspect becomes more and more important as the access time to the memory seems to decrease more slowly than the cycle time of the processor. In addition, it is presented in this paper that there are many more criteria which must be consider when a designer wants to develop a cryptographic system for an application. It could be implemented in software, hardware or a co-design manner, with a variety of algorithms and platforms.

Acknowledgments

The author wishes to thank the anonymous referees for their careful reading of the manuscript and their fruitful comments and suggestions.

References

[1] N. Sklavos, PhD Thesis on *"VLSI Designs of Wireless Communications Security Systems"*, proceedings of 12th International Conference on Very Large Scale Integration, (IFIP VLSI SOC '03), Darmstadt, Germany, December 1-3, 2003.

[2] Eashwar Thiagarajan and Madhuri Gourishetty, "*Study of AES and its Efficient Software Implementation*", Department of Electrical Engineering & Computer Science, Oregon State University, Corvallis, Oregon 97331 -USA, 2001.

[3] Guido Bertoni1, Luca Breveglieri1, Pasqualina Fragneto, Marco Macchetti, and Stefano Marchesin "*Efficient Software Implementation of AES on 32-Bit Platforms"*, in Cryptographic Hardware and Embedded Systems - CHES 2002, pp. 159-171, B.S. Kaliski Jr., .K. Ko, C. Paar.

[4] Chi-Feng Lu, Yan-Shun Kao, Hsia-Ling Chiang, Chung-Huang Yang, "*High Speed Software Driven AES Algorithm on IC Smartcards*", SCIS 2004 The 2004 Symposium on Cryptography and Information Security Sendai, Japan, Jan.27-30, 2004

[5] Andreas Sterbenz, Peter Lipp, "*Performance of the AES Candidate Algorithms in Java*", AES Candidate Conference 2000, pp. 161-165.

[6] Kazumaro Aoki, helger Lipmaa, "*Fast Implementation of AES Candidates*", in submitted for publication-third AES Candidate Conference, New York City, USA, August 13-15, 2001.

[7] Kris Gaj and Pawel Chodowiec, "*Fast implementation and fair comparison of the final Candidates for Advanced Encryption Standard using Field Programmable Gate Arrays*", Proc. RSA Security Conference - Cryptographer's Track San Francisco, CA, April 8-12, 2001.

[8] Weidong Shi, Hsien-Hsin S. Lee, Chenghuai Lu, and Mrinmoy Ghosh, "*Towards the Issues in Architectural Support for Protection of Software Execution*", ACM SIGARCH Computer Architecture News, pp. 6-15, 2005, ISSN: 0163-5964.

[9] Piotr Mroczkowski, *"Implementation of the block cipher Rijndael using Altera FPGA"*, Military University of Technology, Warsaw, Poland, 2000, pmrocz@mamut.isi.wat.waw.pl

[10] N. Sklavos, G. Selimis and O. Koufopavlou, "*FPGA Implementation Cost & Performance Evaluation of IEEE 802.11 Protocol Encryption Security Schemes*", Proceeding of Second Conference on Microelectronics, Microsystems and Nanotechnology, (MMN'04), November 14-17, Athens, Greece 2004.

[11] N.Sklavos, O. Koufopavlou, "Architectures and VLSI Implementations of the AES-Proposal Rijndael", IEEE Transactions On Computers, Vol. 51, No. 12, December 2002.

[12] Hagai Bar-EI, "*Security Implecations of Hardware vs. Software Cryptographic Modules*", http://www.discretix.com, October 2002.

[13] Thomas Wollinger, Christof Paar, *"How Secure are FPGAs in Cryptographic Applications?"* FPL 2003, Vol. 2778, pp.91-100, 2004.

[14] Elisabeth Oswald, Joan Daemen, Vincent Rijmen, "*AES - The State of the Art of Rijndael's Security*", October 30, 2002.