Proceedings of the 2nd IEEE International Conference On Information & Communication Technologies: From Theory To Applications (ICTTA'06), Damascus, Syria, April 24-28, 2006.

Area-Optimized Architectures & Implementation of PELICAN MAC Function

N. Sklavos [I], and C. Efstathiou [II]

[I]: *Electrical & Computer Engineering Dept., University of Patras, Greece, email: nsklavos@ieee.org*

[II]: *Informatics Dept., Technological Educational Institute, TEI of Athens, Greece*

Abstract

Security has become a crucial issue in both communication systems and networks. PELICAN is a latest developed MAC function based on RIJNDAEL encryption algorithm. In this work two alternative architectures, for the efficient VLSI implementation of PELICAN, are proposed. Both architectures design aim to high optimization, regarding requested covered area. Comparisons concerning performance and allocated resources are given, with the conventional design.

1. Introduction

With the advance of both information technology and data communications, security and privacy have been widely used in more area than ever [1]. Yet due to the characteristics of data transmission with no protection over a wired or unwired network, security issues are proven a matter that gains more and more importance everyday. This made privacy an issue of greatest interest for further and future applications, of every communication network and protocol.

In order the special needs for security to be satisfied as better as possible, new security schemes are developed, with main scope to provide a better security level. At the same time, the performance issues of the above designs are also kept in mind, with main goal the most efficient optimization, concerning both speed and area resources.

PELICAN has been proposed as a new MAC function [2]. The design philosophy behind PELICAN is based on the Alred construction, which has been introduced [3]. PELICAN introducers about the security of Alpha-MAC, the concrete design presented in [3], resulted in several suggestions for modifications. The modifications result in a simpler design, because the injection layout map is removed. Secondly, the new design has a slightly better performance.

PELICAN is based on RIJNDAEL [4]. PELICAN instructors have chosen RIJNDAEL mainly because they expect it to be widely available, thanks to its status as the AES standard. Additionally, RIJNDAEL is efficient in hardware and software and it has withstood intense public scrutiny very well since its publication.

This work deals with the implementation of PELICAN function. Especially two different architectures are proposed for the efficient integration of the function. The first one is based on two cores of the RIJNDAEL encryption algorithm, while the second uses only one. In order to have a fair and detailed comparison, the conventional architecture is also presented. With the term conventional is defined the one, that could be developed as a straight forward design from the PELICAN specifications. Both proposed and conventional architectures are compared in terms of both performance and allocated area resources.

This paper is organized as follows: In Section 1 an introduction to the focussed topic is given. In Section 2 a theoretical background regarding the category of hash functions and PELICAN specifications is presented. In Sections 3 & 4, both conventional and proposed architectures are presented in detail. The implementation results for the proposed hardware implementations are given in Section 5. The paper ends with the conclusions which are discussed in Section 6.

2. Hash Functions & PELICAN: A Theoretical Background

2.1. An Overview of Hash Functions Design

Hash functions are widely used for digital signature schemes, data integrity, HMAC and other cryptographic purposes such as random number generators [5], [6], [7]. These functions map data messages of an arbitrary length to a hash value of fixed length. This value is called message digest. There have been many efforts to support hash functions operation on previously designed block ciphers, in order to avoid the construction of a new hash design from scratch [6]. Especially, from the integration point of view, when hardware implementations of an encryption algorithm are available, the cost of the final design could be decreased. This is due to the fact that in this case there exist a block cipher implementation, and with no much additional cost it is efficient to support the operation of such function in the same core.

The main philosophy behind the above design approach is illustrated in the next Figure 1, where the architecture of a hash function round is given [6]:

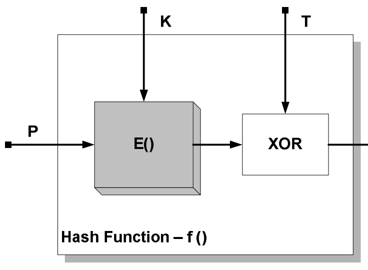

Figure 1: Hash Function Round

E() declares a well known secure block cipher. The input P, which is used as plaintext and the input K, which declares the cipher's key, are combined together in order to produce the output X(i+1). The value of the output round X(i+1) is especially the value of the XORed output between the algorithm's ciphertext and the parameter T.

The input arbitrary length message M, firstly is divided into n blocks M1, …, Mn. The values of P, K, & T, are chosen from the set S = {V, Mi, Xi, Mi *XOR* Xi}, where V is a constant value, Xi is the output of previous round

2.2. PELICAN Hash Function Specifications

PELICAN is an ALRED construction with RIJNDAEL [4], restricted to a block length of 128-bit. Same with RIJNDAEL, PELICAN supports keys of 128-, 160-, 192-, 224- and 256- bit. For the key lengths 128, 24 and 32 bytes, RIJNDAEL coincides with AES. PELICAN can take a message m of any length and generates tags with length up to 128 bits.

PELICAN operation could be distinguished in a number of steps. First the message is padded and split in the message words. These words are applied to a state that is initialized using a key. Afterwards undergoes a final step again using the key. The desired Tag is generated by taking the first l_m bits of the 128-bit, in total, of the state.

3. Conventional Design

The conventional architecture for PELICAN MAC function is illustrated in the following Figure 2. The design of this architecture is based on PELICAN specifications [2], which define two different AES schemes: one "Full Core" of RIJNDAEL and one "Half Core", which performs only four rounds of encryption. It has to be mentioned that the "Half Core" operates with round keys set to zero. All RIJNDAEL cores, Full or Half, work with 128-bit data states.

The basic components of the conventional architecture besides the RIJNDAEL cores are the Padder & Splitting Unit, the XOR block chaining, and the Truncation Unit.

In the Padder & Splitting Unit the message is padded by appending a single 1 bit, followed by the minimum number of zero bits so that the resulting length is a multiple of a vector equal to 128-bit (padding method 2 in [3]). Afterwards, the padded data vector is split n in 128-bit message words X_1, X_2, ..., X_n. The X_i message word serves as the one input of the i XOR Chaining Block.

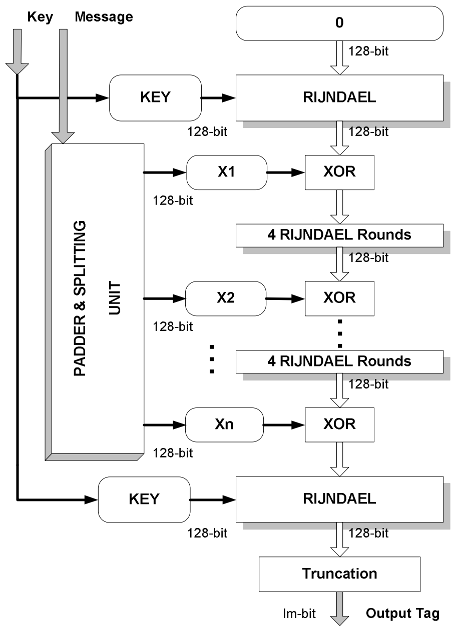

Figure 2: Conventional Design

Based on the conventional design, n message words could be transformed on parallel, and in this way a new Tag (output) is finally produced in each clock cycle.

In the initialization phase, the state is generated as follows: the 128-bit contents of the state are filled with binary zeroes and subsequently the state is applied to RIJNDAEL "Full Core". In the next step the state is encrypted using the input key.

Then, the RIJNDAEL output state is XORed with the first message word X_1. For each additional message word X_i, the Iteration Function is applied to the input state, and then the word X_i each time is XORed with the output state. With the term Iteration Function is declared four RIJNDAEL rounds operation ("Half Core"), with round keys set to zero.

In the final step the output state is applied to the last RIJNDAEL "Full Core" and is encrypted by using the initial key. In the Truncation Unit, the desired Tag is generated by taking the first l_m bits of the 128-bit last state.

4. PELICAN Proposed Architectures

4.1. Proposed PELICAN Architecture: PA2C

The first proposed architecture (PA2C-Proposed Architecture with 2 Cores) for PELICAN function implementation is shown in the next Figure 3. This architecture is based one two different RIJNDAEL cores.

One with full operation named RIJNDAEL and another one, called 4 RIJNDAEL Rounds, which performs only 4 rounds of ciphering. Between the two cores a 128-bit register has been placed, for temporary data storage.

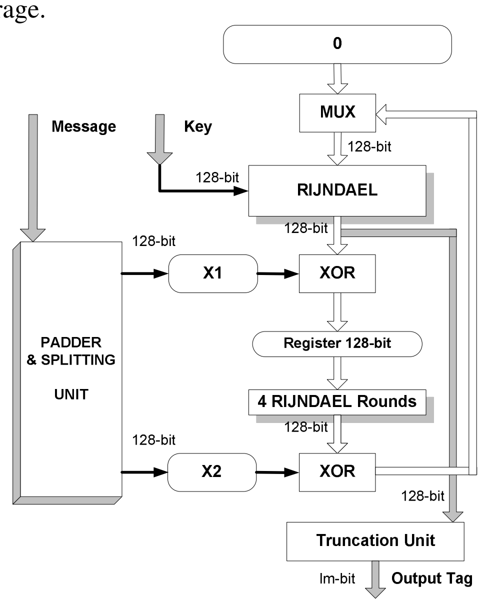

Figure 3: Proposed PELICAN Architecture: PA2C

The proposed architecture uses a similar Padder & Splitting Unit, as the conventional one. This unit is responsible for padding the input message, according to PELICAN specifications, defined by the MAC Hash introducers [2]. The next process of this unit is also to split the padded message in blocks equal to 128-bit.

These blocks are forced through the blocks outputs to the XOR chains. The other input of XORs are the ciphertexts produced by the two RIJNDAEL cores ("Full" and "Half"), of the previous steps.

The output of each XOR feeds a RIJNDAEL core input each time, of the next step operation.

The appropriate number of padded 128-bit blocks, for the initial message is modified in a serial way, through the use of a feedback. For this reason the outputs X_1 and X_2 of the Padder & Splitting Unit are used alternatively. The last padded block is modified always by the RIJNDAEL "Full Core", while the corresponding ciphertext is forced to the Truncation Unit for the appropriate final modifications.

4.2. Proposed PELICAN Architecture: PA1C

The second proposed architecture (PA2C-Proposed Architecture with 1 Core) for PELICAN function implementation is illustrated in the next Figure 4. This architecture uses one core of RIJNDAEL block cipher, which operates, as both "Full" and "Half" Core alternatively.

This is the basic idea, in which PA1C design is based on. In this way the cost of a second RIJNDAEL core is avoided, with major allocated resources benefits as well. Analytical comparison results would be given in the next Section 5, regarding all the presented architectures.

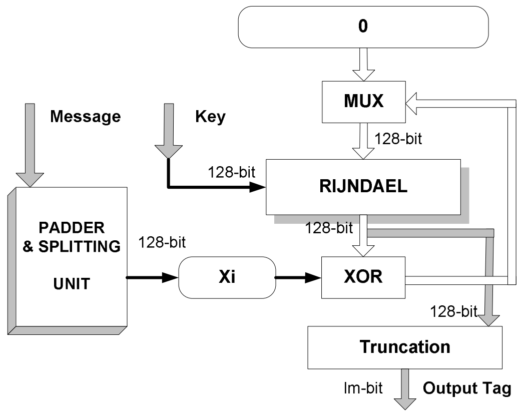

Figure 4: Proposed PELICAN Architecture: PA1C

5. Implementation Results & Comparison

Both conventional and proposed architectures could be implemented with software and also hardware integration platforms. Although, for the integration of encryption algorithms and privacy schemes, for a great number of reasons, regarding both security matters and performance issues, hardware implementations are mostly preferred [8].

It has to be mentioned, that the implementation of the proposed architectures is centered to integration devices on silicon. Although, they could also be developed with software description languages. Such approaches, with software tools, conclude to similar implementation results, concerning performance values and allocated resources requests.

In the next Table I the requested area resources for the VLSI integration of Conventional, PA2C, and PA1C architectures are presented.

Architectures / *Components*	Conventional	PA2C	PA1C
RIJNDAEL Cores	*K*	*2*	*1*
XOR Blocks	*K-1*	*2*	*1*
Register (128-bit)	*K*	*2*	*1*
Padder & Splitting Unit	*1*	*1*	*1*
Truncation Unit	*1*	*1*	*1*
Initial Key Register	*1*	*1*	*1*

Table I: Area Resources

The component with the major area needs is RIJNDAEL core [9]. A representative hardware implementation cost, using an FPGA implementation platform [9] is presented in the following Table II:

RIJNDAEL Architecture Type	FPGA Device (Virtex)	CLB Slices	F (MHz)	Data Rate (Mbps)
Full Rolling	*XCV300 BG432*	*2358*	*22*	*259*
Pipeline	*XCV1000 BG560*	*17314*	*28,5*	*3650*

Table II: RIJNDAEL FPGA Implementation Results

All the other components request low area resources, compared with RIJNDAEL core, which are equal in total to the 10-15% of the whole design. So, the interest regarding the area resources is centered to RIJNDAEL core.

Conventional architecture is proven high resources consumed, especially for greatest values of variable K as it is shown by the graph of Figure 5. In the cases that K is equal to 1 and 2, conventional architecture needs almost the same resources with proposed PA1C and PA2C respectively.

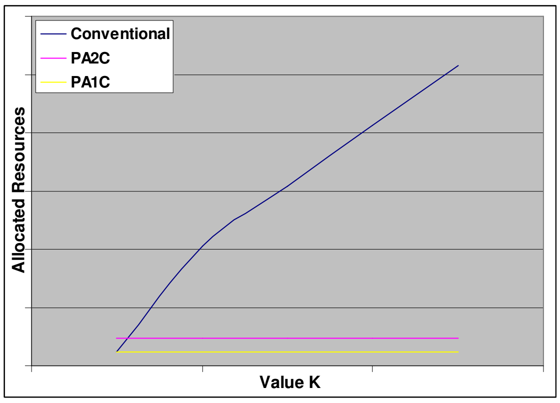

Figure 5: Area Comparison

Despite of the fact that Conventional architecture is really proven high consumed, regarding area allocation, the performance of such architecture is very high. This is due to the fact that Conventional transforms K padded data vectors in parallel. This results to a new produced Tag in every clock cycle. In other words, K data words are completely modified in K clock cycles.

On the other hand PA2C is based on two RIJNDAEL cores and needs (K+1) clock cycles, in order two new Tags to be produced.

Furthermore, PA1C proposed architecture has been designed with one RIJNDAEL core and transforms 1 padded data work per K clock cycles.

The following graph of Figure 6 is representative of performance, regarding the three architectures. It has to be mentioned that for all architectures the valued of frequency is the same. This is due to the fact, that the time critical path of each architecture, is equal to RIJNDAEL Core delay time (t_{delay}).

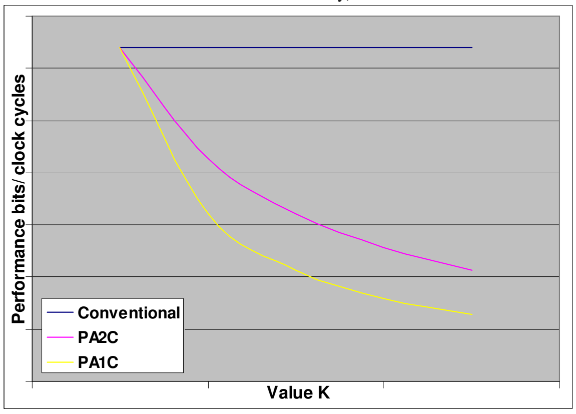

Figure 6: Performance Comparison

6. Outlook

With the advance of both information technology and data communications, security and privacy have been widely used in more area than ever. PELICAN has been proposed as a new MAC function. This work deals with the implementation of PELICAN function. Especially two different architectures are proposed for the efficient integration of the function. In order to have a fair and detailed comparison, the conventional architecture is also presented.

7. Acknowledgment

This work is co-funded by 75% from the E.E., and 25% from the Greek Government under the framework of the Education and Initial Vocational Training Program-Archimedes.

8. References

[1] N. Sklavos, and O. Koufopavlou, "Mobile Communications World: Security Implementations Aspects - A State of the Art", CSJM Journal, Institute of Mathematics and Computer Science, Vol. 11, Number 2 (32), pp. 168-187, 2003.

[2] J. Daemen and V. Rijmen, "The Pelican MAC Function", Cryptology ePrint Archive, 088/2005.

[3] J. Daemen and V. Rijmen, "A new MAC Construction Alred and a Specific Instance Alpha-MAC,", *Fast Software Encryption 2005, LNCS* H. Gilbert, H. Handschuh, Eds., Springer-Verlag, to appear.

[4] *Federal Information Processing Standard 197, Advanced Encryption Standard (AES)*, NIST, U.S. Department of Commerce, November 2001.

[5] Bruce Schneier, Applied Cryptography – Protocols, Algorithms and Source Code in C, Second Edition, John Wiley and Sons, New York, 1996.

[6] S. Bakhtiari, R.Safavi-Naini, J. Pieprzyk, "Cryptographic Hash Functions: A Survey", Technical Report 95-09, Department of Computer Science, University of Wollongong, July 1995.

[7] *ISO/IEC 9797-1, Information technology - Security Techniques - Message Authentication Codes (MACs) - Part 1: Mechanisms using a block cipher,* ISO 1999.

[8] N. Sklavos, K. Touliou, and C. Efstathiou, "Exploiting Cryptographic Architectures over Hardware Vs. Software Implementations: Advantages and Trade-Offs", proceedings of the 5th International Conference on Applications of Electrical Engineering (AEE '06), Prague, Czech Republic, March 12-14, 2006.

[9] N. Sklavos and O. Koufopavlou, "Architectures and VLSI Implementations of the AES-Proposal Rijndael", IEEE Transactions on Computers, Vol. 51, Issue 12, pp. 1454-1459, 2002.