

# Design and Hardware Implementation of QoS-AES Processor for Multimedia applications

Zeghid Medien\*\*, Mohsen Machhout\*, Belgacem Bouallegue\*, Lazhar Khriji\*\*\*, Adel Baganne\*\*, Rached Tourki\*

\* Electronic and Micro-Electronic Laboratory, Faculty of Sciences of Monastir, 5000, Tunisia.

E-mail: medien.zeghid@fsm.rnu.tn

\*\* Laboratoire des Sciences et Techniques de l'Information, de la Communication et de la Connaissance (Lab-STICC), CNRS: FRE2734 –University of South Brittany, Lorient, France.

E-mail: adel.baganne@univ-ubs.fr

\*\*\* ATSI- Research Unit, ISSATSO, University of Sousse, Tunisia.

**Abstract.** For real-time applications, there are several factors (time, cost, power) that are moving security considerations from a function centric perspective into a system architecture (hardware/software) design issue. Advanced Encryption Standard (AES) is used nowadays extensively in many network and multimedia applications to address security issues. The AES algorithm specifies three key sizes: 128, 192 and 256 bits offering different levels of security. To deal with the amount of application and intensive computation given by security mechanisms, we define and develop a QoS (Quality of Security Service) model for reconfigurable AES processor. QoS has been designed and implemented to achieve a flexible trade-off between overheads caused by security services and system performance. The proposed architecture can provide up to 12 AES block cipher schemes within a reasonable hardware cost. We envisage a security vector in a fully functional QoS request to include levels of service for the range of security service and mechanisms. Our unified hardware can run both the original AES algorithm and the extended AES algorithm (QoS-AES). A novel on-the-fly AES encryption/decryption design is also proposed for 128-, 192-, and 256-bit keys.

The performance of the proposed processor has been analyzed in an MPEG4 video compression standard. The results revealed that the QoS-AES processor is well suited to provide high security communication with low latencies. In our implementation based on Xilinx Virtex FPGAs, speed/area/power results from these processors are analyzed and shown to compare favorably with other well known FPGA based implementations.

---

## 1 Introduction

Security issues involved in embedded systems ranging from low-end systems such as networked sensors, wireless handsets, PDAs, and smart cards to high-end systems such as routers, gateways, firewalls, storage servers, and web server become more and more important. For such systems, there are several factors governing security from a function centric perspective to a system architecture (hardware/software) design issue. For example: The processing capabilities of many embedded systems are easily overwhelmed by the computational demands of security processing, leading to undesirable tradeoffs between security and cost, or security and performance. Battery-driven systems and small form-factor devices such as PDAs, cell phones and networked sensors often operate under stringent resource constraints (limited battery, storage and computation capacities). These constraints go worse when the device is subject to the demand of security like real-time multimedia applications that will impose requirements for time. Embedded system architectures need to be flexible enough to support the rapid growth of security mechanisms and standards. Cryptographic Algorithms that require increasing computation capabilities are developed such as AES.

In November 2001, the National Institute of Standards and Technology (NIST) of the United States chose the Rijndael algorithm as the suitable AES [3] to replace the Data Encryption Standard (DES) algorithm [1]. There are three kinds of choice for the cipher key of the AES: 128-, 192- and 256-bit, called AES-128, AES-192, and AES-256, respectively [3]. AES has 10 rounds for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys. The design and strength of all key lengths of the AES algorithm are sufficient to protect classified information up to the SECRET level. AES provides three levels of security: 128, 192, and 256 bits. AES, very low memory requirements, make it very well suited for restricted-space environments, for which the AES exhibits high performance. The AES algorithm is used in some applications such as smart cards, cellular phones and image-video encryption. The efficiency of hardware implementations of the AES has been used by the designer as major criterion.

Since then, many hardware implementations have been proposed in literature [12-24]. Some of them use Field Programmable Gate Arrays (FPGA) and others use Application-Specific Integrated Circuits (ASIC). Despite the advantage of a software implementation that includes ease of use, ease of upgrade, portability, and flexibility, it suffers from its limited physical security, especially with respect to key storage. Conversely, cryptographic algorithms (and their associated keys) implemented in hardware are, by nature, more physically secure, as they cannot easily be read or modified by an outside attacker. Reconfigurable hardware devices such as FPGAs are promising alternatives for the implementation

of block ciphers. FPGAs are hardware devices whose functions are not fixed and can be programmed in-system.

To deal with the amount of system performance and intensive computation given by security mechanisms, this paper provides a motivation for understanding QoS and variant security for AES, and how these concepts may benefit variant applications and systems designs. So, we define and we develop a QoS model for reconfigurable AES processor. The proposed architecture can provide up to 12 AES block cipher schemes within a reasonable hardware cost. Using a security vector, the user can assign a precise security service of his application (cryptographic operation, mode, key-length, rounds number, quality of key, throughput). Our unified hardware architecture can run the original AES algorithm and the extended QoS-AES algorithm as well. Furthermore, this paper proposes a video encryption scheme, which includes the QoS-AES processor. Such a processor encrypts the DC DCT (discrete cosines transform) transform coefficients of each block in digital video. Since DC coefficients play an important role in MPEG video data, the method is designed to provide relatively high level of security for DC coefficients.

The paper is structured as follows: The next section details the proposed architecture and the overall design of the AES implementation. Also presented are the area and timing. Issues related to security levels of the cryptosystems considered are discussed in section 3. The QoS module is defined as well. Section 4 explains the details of our design on the QoS-AES cryptographic processor and compares the performance of our design to earlier ones. The illustration of the usefulness and the efficiency of the proposed processor through communication examples using an MPEG4 decoder are described in Section 5. Finally, concluding remarks are made in Section 6.

## 2 Choice of an Architecture for AES Processor

### 2.1 Our Choice

The choice of a suitable architecture has a significant impact on system performance. Different metrics can be used for this purpose; such as throughput, power consumption, area, security levels, resistance to side channel attacks, synchronous or asynchronous architecture and modes of operation. To this end, our objective is to define appropriate architecture for the AES processor. So far, the basic techniques for implementing a block cipher with rounds are iterated, pipelined, and loop-unrolled architectures. The iterated architecture leads to the smallest implementation. Such architecture consists in a round component, which is fed to its output. The pipelined architecture contains all of the rounds as separated components. As a result, it is the fastest in terms of throughput and

the largest of the basic structures. The loop-unrolled architectures achieve two or more rounds per clock cycle and the execution of the cipher is iterated. In a pipelined architecture, unrolling can only decrease the latency of outputting the first block. In sub-pipelining, registers are placed inside the round component in order to increase the maximum clock frequency. In the partial pipelining scheme, the pipeline contains the half of the rounds with registers in between. So, pipelined and loop-unrolled architectures enable very high-speed implementations; but they imply large area and high power consumption. This fact makes them unattractive for embedded systems, such as smart cards. Furthermore, they cannot be fully exploited in feedback modes of operation. Feedback modes are often used for security reasons in encryption and for Message Authentication Code (MAC) generation. In feedback modes, like Cipher Block Chaining (CBC) and Cipher Feedback (CFB) mode, it is not possible to start by encrypting the next block of data before the encryption of the previous block is completed. As a result, all blocks must be encrypted sequentially, with no capability for parallel processing. In the non-feedback modes, like Electronic Code Book (ECB) mode and counter (CTR) mode, encryption of each subsequent block of data can be performed independently from processing other blocks, thus all blocks can be encrypted in parallel.

The basic iterative architecture assures the maximum speed/area/ratio for feedback operating modes (CBC, CFB). It is commonly used for bulk data encryption and, also, to guarantee near optimum speed, and near optimum area for these operating modes. The width of the AES data path can be further reduced to decrease logic area and power. Hence, we chose the basic synchronous iterative architecture in our implementation. The subsection below presents the details of the design.

## 2.2 AES implementation using an iterative design

Figure 1 shows the block diagram of the different units of the AES-128 based processor.

As seen, the AES-128 core includes the following units.

- The Input and Output interfaces take care of reading input data and writing encrypted output and are responsible for feeding the key logic. They are controlled by the data-ready, ciphertext-ready and clk signals. When the bus puts a data to be read or write this signal is selected and the data is taken.
- Library Functions: the different AES functions in this library, such as SubBytes, inv-SubBytes, ShiftRows, inv-ShiftRows, MixColumns, inv-MixColumns and AddRoundkey are defined.
- Sbox ROM: is the direct implementation of the substitution boxes using lookup tables, because all of the 256 cases of the substitution bytes can be pre-computed and can be stored in a lookup table.

- RAM keys: are used to load the initial key to the FPGA through the input interface.
- Key Expander is used to compute a set of round keys based on a ciphered key.
- Controller is used to generate control signals for all other units. Among other actions, the controller determines when to reset the cipher hardware, to accept input data and to register output results. Since the execution of MixColumn function is conditional (Figure 2), the controller decides whether the result obtained by MixColumn is to be used or ignored.
- AES-Round, used to encrypt or decrypt input blocks of data.

In our application, ECB/CBC mode of operations is employed for data confidentiality and authentication.

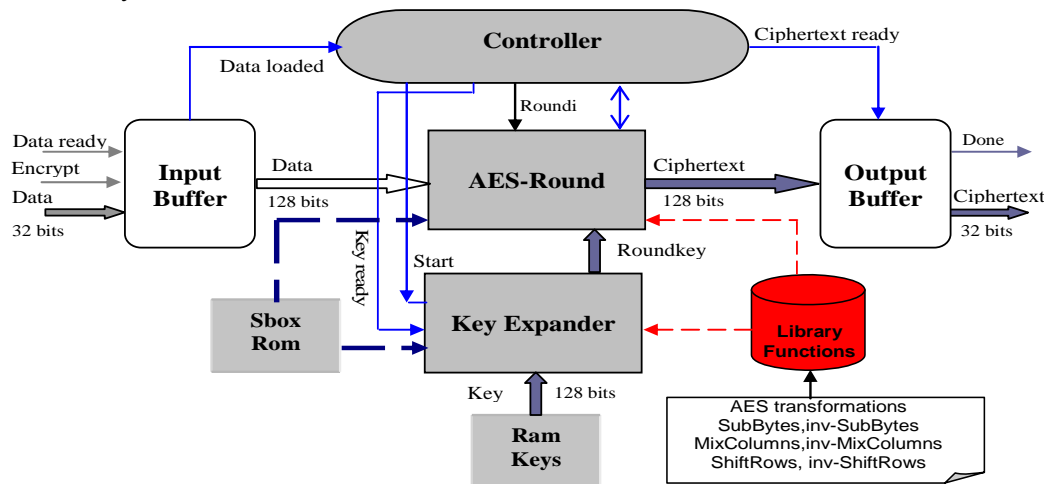


Figure 1. Iterative design of the AES-128 processor

### 2.2.1 Proposed Architecture

The encryption-decryption datapath of the iterative AES-128 core is based on the single round implementation of the AES algorithm. The same datapath is used for the 10 rounds in the AES-128 algorithm. Each round is performed in a single clock cycle except the first round (round zero) that only performs the key addition phase. The other 10 rounds perform the four different steps of the AES algorithm namely SubBytes, Shiftrows, MixColumns (not done last round), and AddRoundKey. Therefore, it takes total of 11 cycles to encrypt or decrypt a 128-bit block of data. As shown in figure 2, five logic blocks compose the overall AES Round.

- The component implementing the function AddRoundKey is simply a net of XOR gates that adds in  $GF(2^8)$  the round key to the current state.
- The component implementing the function SubBytes uses 16 S-boxes stored in a Read-Only Memory (ROM). The state obtained is row-shifted.

- The MixColumns function is implemented using a chain of XORs which results in the minimum delay implementation for this unit.
- The AES-128 encryption and decryption data path is optimized to have a minimum delay for each round.

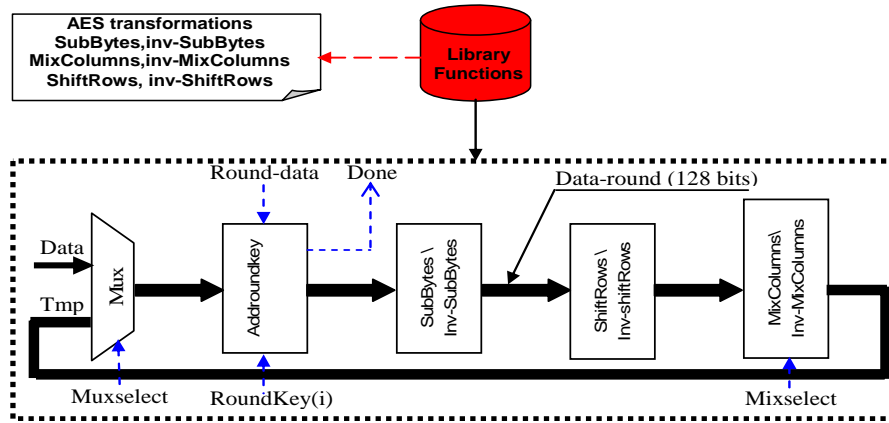


Figure 2. AES Round block implementation

Overall, the total combinational delay of all the four steps of the AES-128 are minimized so that each round of the AES-128 algorithm is performed in a single clock cycle at a maximum clock frequency.

The decryption process follows the same order as the encryption, except for another round of mixed columns on the generated round key.

## 2.2.2 Experimental Results

We have implemented AES using the technique mentioned above. The AES-128 encryption/decryption system is captured using VHDL. The Xilinx ISE tools have been used for the implementation of the design and the simulation environment is Modelsim. Xilinx XC2V1000 has been chosen as the target device. The architecture is simulated to verify the functionality, with use of the test vectors provided by the AES-standard [3]. In order to have a fair and detailed evaluation, we implemented AES-128 encryption separately. Performance metrics such as area, throughput, and power consumption are used.

Table 1 presents detailed results for these implementations. The process engine is able to operate in feedback mode at 159 MHz, which equates to 1.850 Gb/s. During self-test at 50 Mhz, the process consumed 22.7mW.

Table 2 compares our implementations with recent works reported in literature in terms of AES-128 encryption only, AES-128 decryption only, both encryption and decryption [17], [24]. Comparisons with AES ASICs implementations are also given [12], [13], [14]. As is shown, the throughput of our implementation is better compared to that reported in [17]-[18]-[19]-[20]-[23]-[24], but is lower than that in [21].

**Table 1. Results of the iterative AES-128 design**

Design	Performance metrics			
	Freq (Mhz)	Area (slices)	Power (mW)	Throughput (Mbps)
AES-128-encryption XC2V1000	159	1743	22,7	1850
AES-128-encry/decryp XC2V1000	82	3555	37,50	1049

**Table 2. Performance Comparison Results**

Reference	Architecture	Area	Frequency (MHz)	Throughput (Mbps)
AES-128: ASIC Implementation				
[12]	Non pipelined integrated	173k gates	154	2290
[13]	Non pipelined Decryption	58,43k gates	200	2008
[14]	Non pipelined Encry/Decry	200,5K gates	66	844.8 (AES-128) 704 (AES-192)
AES-128: FPGA Implementation				
[17] XCV300	Non pipelined Encry/Decry	2358 CLB	22	259
[18] XC2V1000	Non pipelined Encry/Decry	4325	75	739
[19] XCV1000E	Non pipelined Encry/Decry	N/A	38,8	451,5
[20] XCV812	Non pipelined Encryption	2744	20,192	258,5
[21] XC2V1000	Non pipelined Encryption	1122	159	1941
[22] Virtex-II	pipelined Encryption	1937	182	2118
[23] Virtex-II Pro	Non pipelined Encryption	2703 LUT + 44 BRAM	196	1190
[24] XC2V1000	Non pipelined Encryption	586 slices + 10 BRAM	96,42	1450
Proposed architectures				
Our works XC2V1000	Non pipelined Encryption	1743	159	1850
Our works XC2V1000	Non pipelined Encry/Decry	3555	82	1049

### 3 Quality of Security Service Requests

Quality of Service (QoS) mechanisms benefits both the user and the overall distributed system. QoS users profit by having reliable access to services. The distributed systems whose resources are QoS managed present the advantage to have more predictable resource utilization and more efficient resource allocation. Also, QoS involves user requests for levels of services which are related to performance-sensitive variables in an underlying distributed system. To include the security as a real part of QoS, the security choices should be presented to users, and the QoS mechanism must be able to modulate related variables as well. References to security in the QoS literature are reported in [25]-[26].

The term Quality of Security Service (QoSS) refers to the use of security as a quality of service dimension. To recap, the enabling technology for both QoSS and a security-adaptable infrastructure is the variant security. It can be considered as the ability of security mechanisms and services to allow the amount (i.e. kind or degree) of security to vary within predefined ranges.

In the following an analysis of the security in the designed AES processor will be presented.

#### 3.1 Multilevel Security

The information is classified into different levels of trust and sensitivity. These levels represent the well-known security classifications, namely, Low, Medium, and High security. They form a simple hierarchy where the data flows from Low level to High level, as illustrated in figure 3.

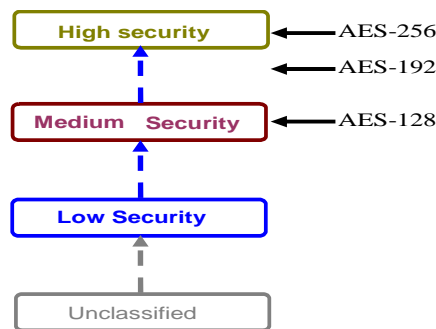


Figure 3. The hierarchical security levels

The U.S National Security Agency (NSA) has conducted a review of the AES encryption algorithm and its applicability to protect national security information. In June 2003, the NSA review determined that the design and the strength of all three AES key lengths are sufficient to protect classified U.S government information up to the 'SECRET' level. Their review concluded that

only the AES 256-bit key length is strong enough to protect classified information at the “High security” level (top level security). Furthermore, the key length in AES could be used as a variable quality of security service. The QoS function is given by,

$$QoS = f(\text{key} - \text{length}) \quad (1)$$

### 3.2 Rounds Number Analysis

The presence or absence of shortcut attacks for a cipher is still considered as a quality criterion that is widely accepted in the cryptographic community. Furthermore, the resistance of iterative block ciphers with respect to a specific cryptanalytic method can be evaluated by performing attack on reduced-round versions of the block cipher. Such attacks can provide more information on the security margin of a cipher. If, for R rounds cipher, there is a shortcut attack against a reduced-round r ( $0 < r < R$ ), the cipher has an absolute security margin of R-r rounds or a relative security margin of (R-r)/R. In fact, the security margin indicates the resistance of the cipher against improvements of known types of cryptanalysis.

Table 3 summarizes attacks including the name of the attack as well as the number of the AES rounds for a given key size. We mention that there are other attacks such as side channel attacks [9]-[11].

**Table3. Shortcut Attacks on Reduced Versions of AES**

Attack	AES-128 10 Rounds	AES-192 12 Rounds	AES-256 14 Rounds	Authors
Impossible differential	6 rounds	7 rounds	7 rounds	[5]
Collision attack	7 rounds	8 rounds	7 rounds	[6]
Square attack	7 rounds	7 rounds	9 rounds	[7]
Partial sums	7 rounds	8 rounds	8 rounds	[8]
Related-key attack	- rounds	- rounds	9 rounds	[8]

In other hand, the number of rounds carried out in the AES processor can be used as a second criterion to characterize the quality of security service. Noticing that more rounds consume more resources and thus provide more security. Then, the new expression of the QoS model is,

$$QoS = f(\text{key} - \text{length}, \text{round} - \text{numbers}) \quad (2)$$

Therefore, consider the security levels definition; the AES possesses 2 rounds of absolute security margin for AES-128, three for AES-192 and four rounds for AES-256. Figure 4 shows the hierarchical security levels for AES.

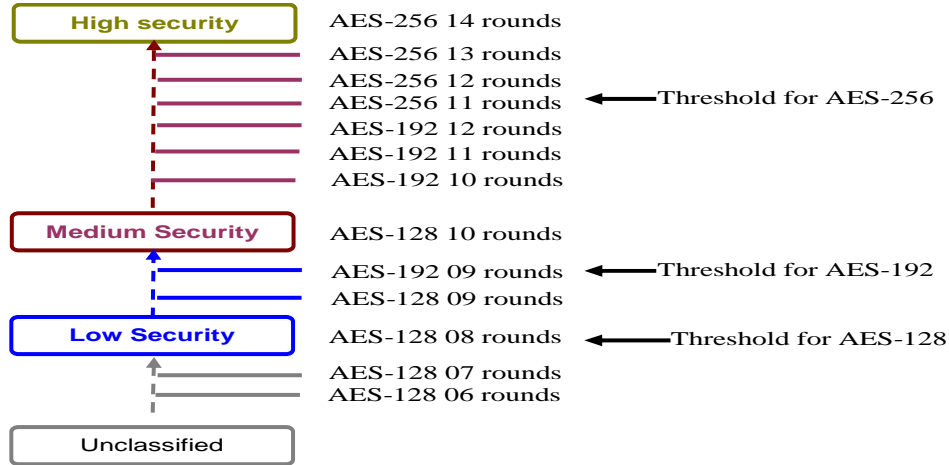


Figure 4. The hierarchical security levels for AES

### 3.3 Key Space Analysis

A convenient encryption scheme is sensitive to the secret keys, and the key space should be large enough to make brute-force attacks infeasible. In the case of the present study, the key space size is  $2^{128}$ ,  $2^{192}$ , and  $2^{256}$  for AES-128, AES-192 and AES-256, respectively. Experimental results demonstrate that AES is very sensitive to the secret key as well. This is shown by a test on the correlation of adjacent pixels in the encrypted image.

We have tested the correlation between two vertically adjacent pixels, and two horizontally adjacent pixels, in an encrypted image. Obtained results are reported in Figures 5 and 6. First, we have randomly selected  $n$  pairs of two adjacent pixels from the image and we have calculated the correlation coefficient of each pair according to:

$$Cov(x, y) = E[(x - E(x))(y - E(y))] \quad (3)$$

Where  $E[.]$  is the mathematical expectation,  $x$  and  $y$  are grey-scale values of two adjacent pixels of the image. Figs.5-6 depict the simulation results of the correlation coefficients of two adjacent pixels in two images (encrypted Lena and Clown test images) using different secret keys,  $K_i$ . As can be seen, when  $K_i$  changes slightly, the encrypted image becomes absolutely different.

By studying the strength of the confusion and diffusion properties, and the security against statistical attack, AES ensures a high security for encrypted image. In Electronic CodeBook (ECB) mode; ciphered block is a function of the corresponding plaintext block, the algorithm and the secret key. Consequently, the same data will be ciphered to the same value; which is the main security

weakness of this mode and the image scheme encryption. In fact, if the image contains homogeneous zones, all the same blocks remain identical after ciphering. In such a case, encrypted image contains textured zones and the entropy of the image is not maximal [27]. Moreover, the quality of the key represents another variable of the quality of security service function. The QoSS function read as:

$$QoSS = f(key - length, round - numbers, key - quality) \tag{4}$$

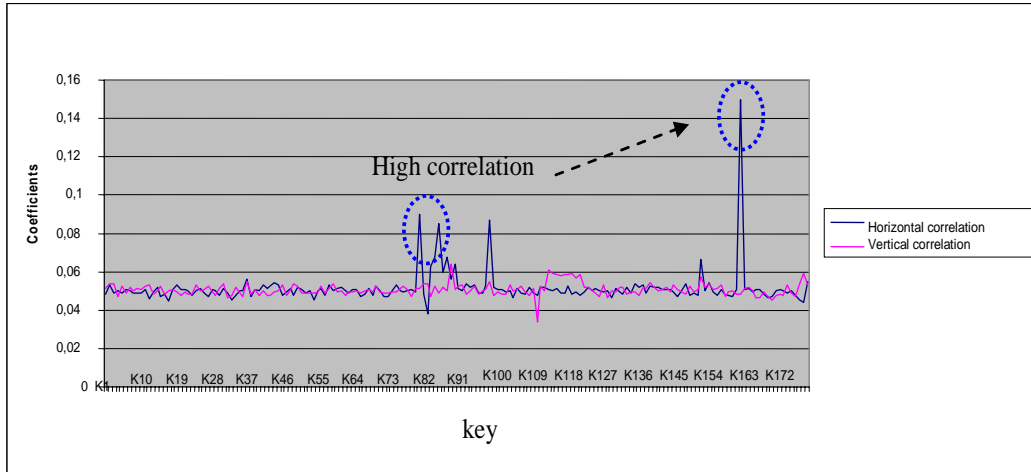


Figure 5. Correlation coefficients of adjacent pixels in the Lena encrypted image using different Ki

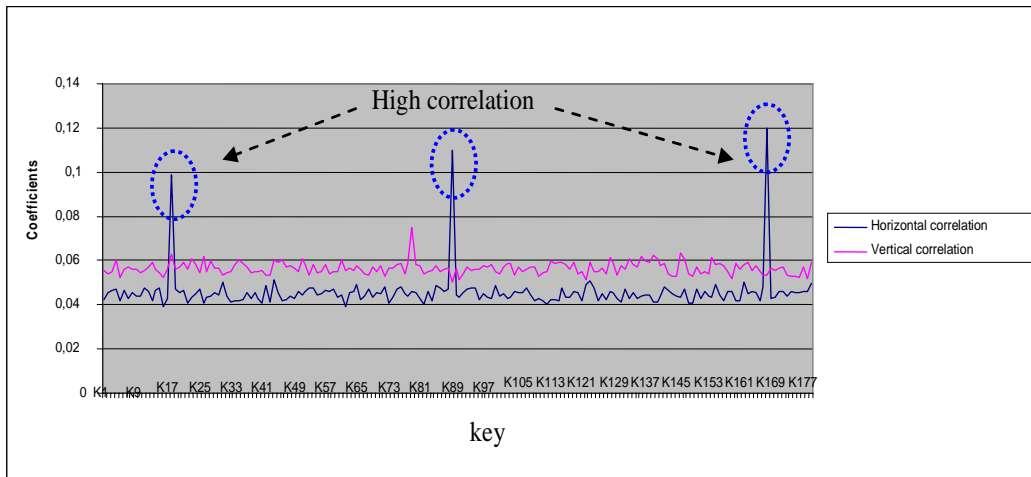
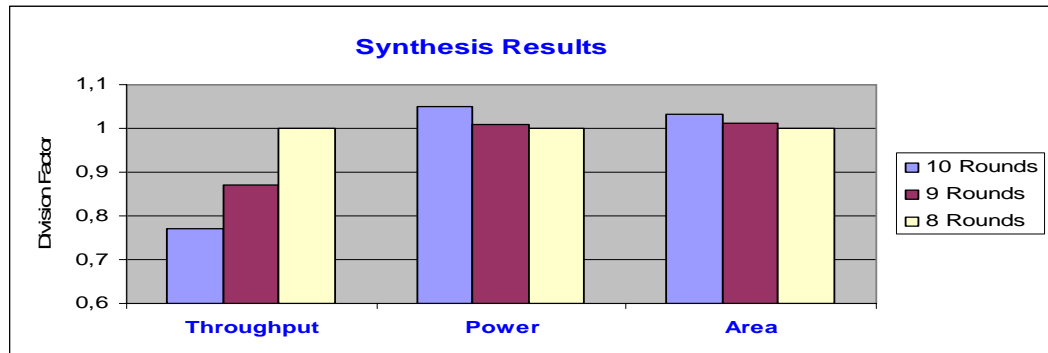


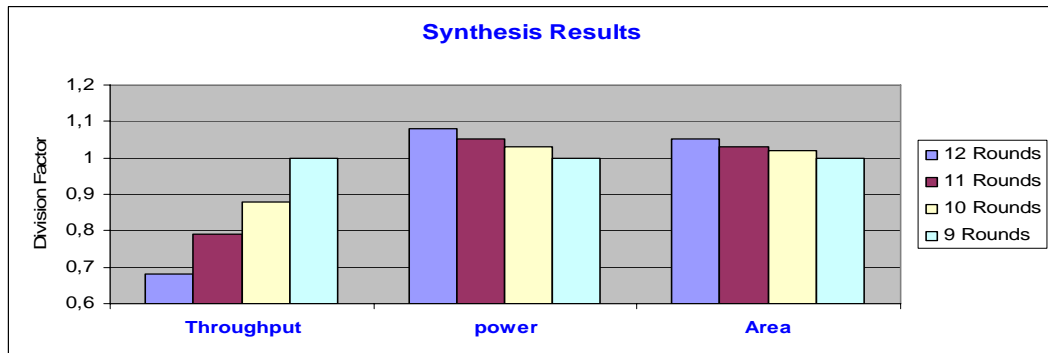
Figure 6. Correlation coefficients of adjacent pixels in the Clown ciphered image using different Ki

### 3.4 Area, Throughput, and Power Analysis

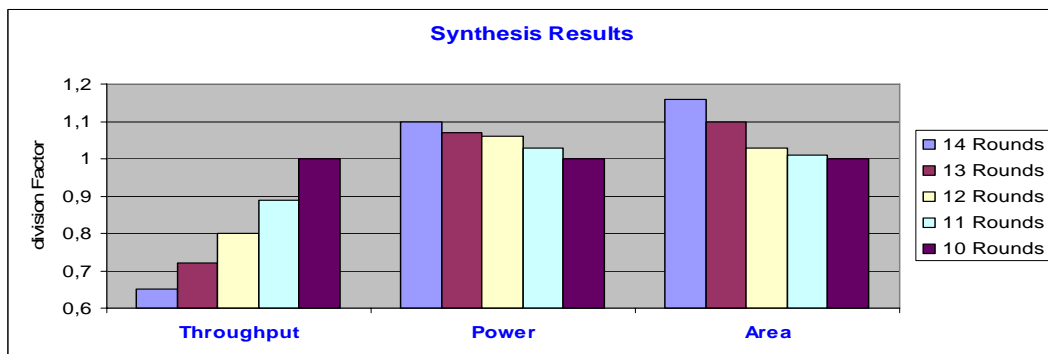
Figure 7 shows the performance of AES-128 (Figure 7-a), AES-192 (Figure 7-b), and AES-256 (Figure 7-c) in terms of area, throughput and power, respectively.



a- AES-128



b- AES-192



c- AES-256

Figure 7. Rounds sensitive tests: a-AES-128, b-AES-192, c-AES-256

As can be deduced from Figure 7 (a-c):

- Implementations of all AES ciphers schemes (AES-128, AES-192, AES-256) ranging from 3540 (8 rounds for AES-128) to 4391 (14 rounds for AES-256) of the total number of CLB slices available in the Virtex XC2V1000 device have been used in our design. This means that, when the number of rounds is varies from  $N_{min} = 8$  to  $N_{max} = 14$ , the area doesn't show a significant change.
- The power consumption changes slightly as well (from 37.50mW for AES-128 10 rounds to 39.30mW for AES-192 12 rounds).

- As also shown, the throughput enhances as the AES round decreases and vice-versa. As a result, the throughput goes from 1366 Mbps for AES-128 (8 rounds) to 654 Mbps for AES-256 (14 rounds).

The figure 8 shows the throughput versus the rounds number. It is easy to see that the AES-128 (8 rounds) performs the largest amount of throughput, it's two times faster than AES-256 (14 rounds). However, the area and the power remain the same as for AES-128 (8 rounds) and AES-192 (8 rounds). In addition, for  $N_r = 10$ , AES-128 and AES-192 have the same throughput. As a consequence, the throughput may also be used to define the QoSS function.

$$QoSS = f(\text{key-length, round-numbers, key-quality, throughput}) \quad (5)$$

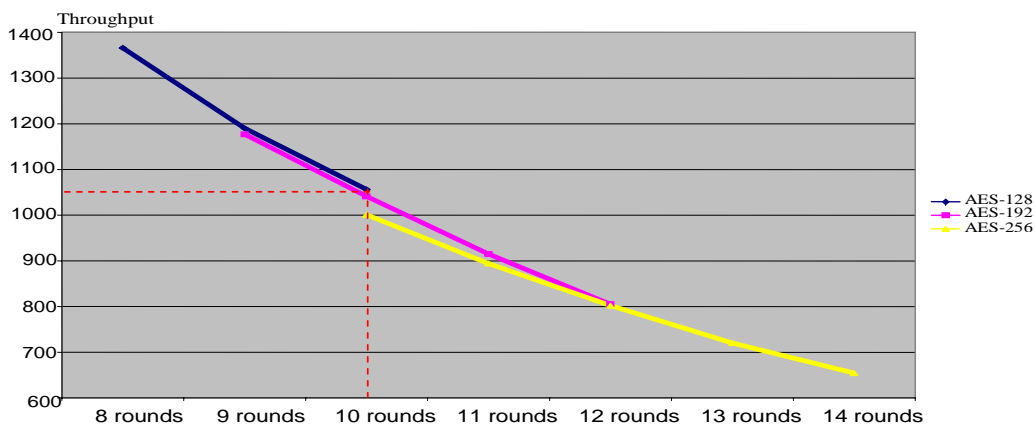


Figure 8. The throughput trade-off of the AES processor

AES-varieties can be divided into two groups: (i) First group requires the medium throughput and the largest security; (ii) Second group requires the largest throughput and the medium security (see Figure 9).

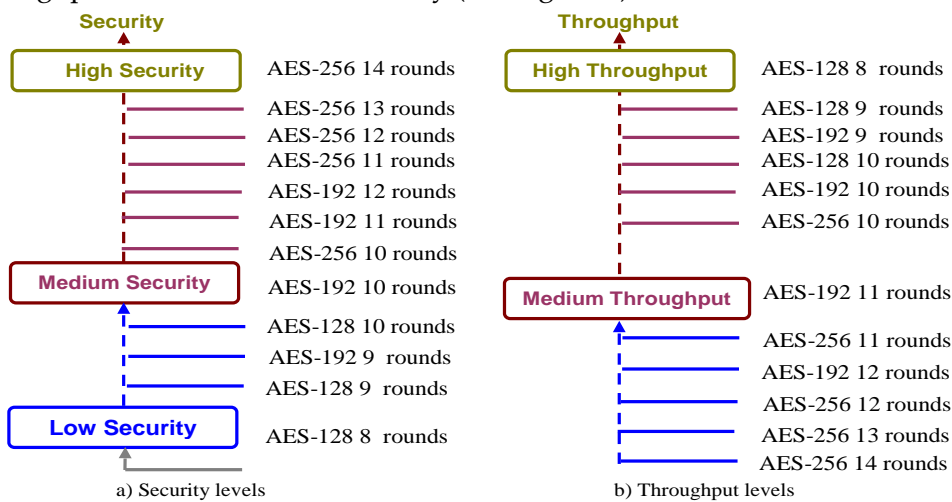


Figure 9. Sensitivity performance ((a) security levels, (b) throughput levels) of AES processor

## 4 QoSS-AES Processor Design

### 4.1 QoSS-AES Implementation using an Iterative Design

In this section we present the architectural design of the QoSS-AES processor. The top-level view of the QoSS-AES encryption decryption processor for an iterative design is shown in Figure 10. The goal of this implementation is to provide up to 12 kinds of extended AES algorithm with the key size of 128, 192 and 256 bits. The original AES algorithm is also included, which supports the ECB and the CBC modes. It includes the input module, AES module, QoSS module and the output module.

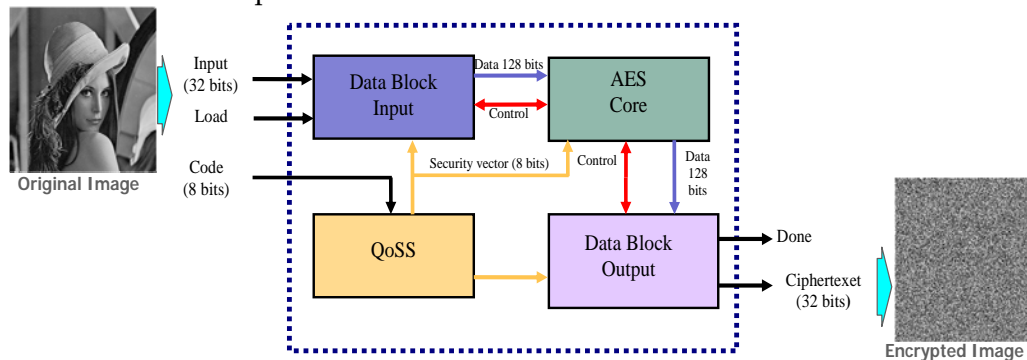


Figure 10. Design of the QoSS-AES processor

The input and output modules perform the handshaking to read the input and to write the encrypted or decrypted data. The main task of these modules is to read or write the 128-bit blocks of data from the 32-bit interface. The AES core is based on the architecture that presented in figure 1. The QoSS module is designed to configure the AES processor, as well as controlling the encryption or decryption protocol. The flow of data through the QoSS-AES processor is transferred via the AES-round module and the output module data paths. The controller takes care of reading the instructions. The QoSS module manages the tasks of different blocs. Following this methodology, the QoSS-AES processor can be configured to encrypt and decrypt the stream of input data with 12 procedures: from AES-128 (8 rounds) to AES-256 (14 rounds).

### 4.2 The QoSS Module

The Quality of Security Service is a module needed to optimize applications security requirements based on a variable system resources. Conceptually, it is an engine or security-critical real-time system to achieve a high performance for the system in terms of quality of security.

The input of the QoSS module is a security service attribute-value specified by users. The output is an array of selective values for each security service re-

quired. The most important abstraction in our QoS module is security levels, which is used to indicate how strength is a particular security service?

Users can define their security requirements for a particular security service by specifying a security range. Figure 11 illustrates the structure of the security vector, which consists of 8 bits. The first bit defines the category of the cryptographic operation (encryption or decryption). Subsequently, there are 2 bits of cryptographic modes. Currently, ECB and CBC modes are implemented in the processor. Also, an extension to other cryptographic modes is expected. Two other fields in the security vector are indicated by Key Pointer (KP) and Round Pointer (RP). They refer to the length of the key and the round numbers, respectively. As depicted in Figure 11(b), the control rounds identify the encryption or decryption scheme. The last bit of the vector is the key address. When the key remains unchanged, the key Ram session can be used for the ciphered key.

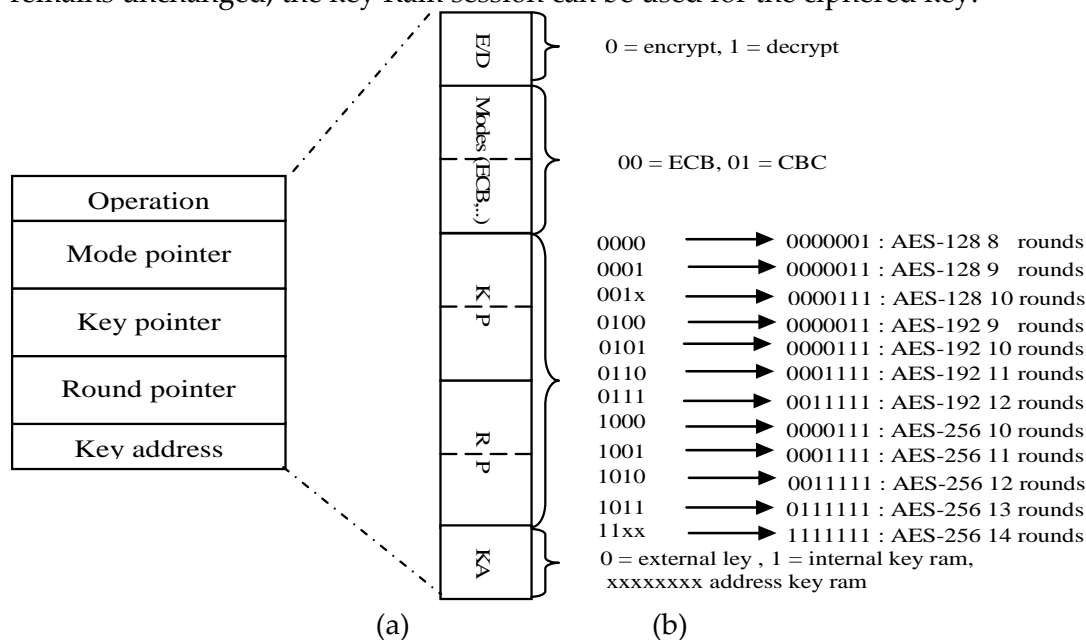


Figure 11. Security vector: (a) General format; (b) Selective Values

### 4.3 Implementation Results

We implemented an FPGA design that, efficiently, performs both AES standard (AES-128, AES-192, and AES-256) and extended AES as known by QoS-AES processor. The proposed processor has been implemented in VHDL using the Model Technology Modelsim simulator and synthesized, placed, and routed using target device of Xilinx (Xilinx Virtex XC2V1000 FPGA). In order to have a fair and detailed evaluation, we implemented AES-128, AES-192, and AES-256, separately. Four performance metrics such as clocking frequency (Mhz), the throughput (Mbps), the area (slices), and the power consumption have been computed. The results of the FPGA implementations are listed in Table 4.

Table4. FPGA Synthesis results

Design	Performance metrics			
	Freq (Mhz)	Area (slices)	Power (mW)	Throughput (Mbps)
AES-128-processor XC2V1000	82	3555	37.50	1049
AES-192-processor XC2V1000	75	3850	39.30	800
AES-256-processor XC2V1000	72	4391	43.91	658
QoSS-AES-processor XC2V1000	71	4470	45	[649, 1136]

It is clear from table 4 that the proposed processor has almost the same allocated resources as the separate AES-256. The processor engine is able to operate at 71 Mhz, which is identical to the frequency of AES-256 and about 14 % less than the 82 Mhz of the AES-128 implementation. Furthermore, during self test at 50 Mhz, the QoSS-AES processor consumed 45 mW. The separate implementations of AES-128, AES-192 and AES-256 have estimated the power dissipation of 37.50mW, 39.30mW and 43.91 mW, respectively.

## 5 Application

This section presents the experimental results that are carried out to evaluate the performance of the QoSS-AES processor in the case of an MPEG4 decoder. A comparative study has been done between the proposed QoSS-AES processor and the conventional video encryption schemes (Sub band Shuffle, Block Shuffle). The results demonstrate that the QoSS-AES processor is well suited to provide high security with very low latency.

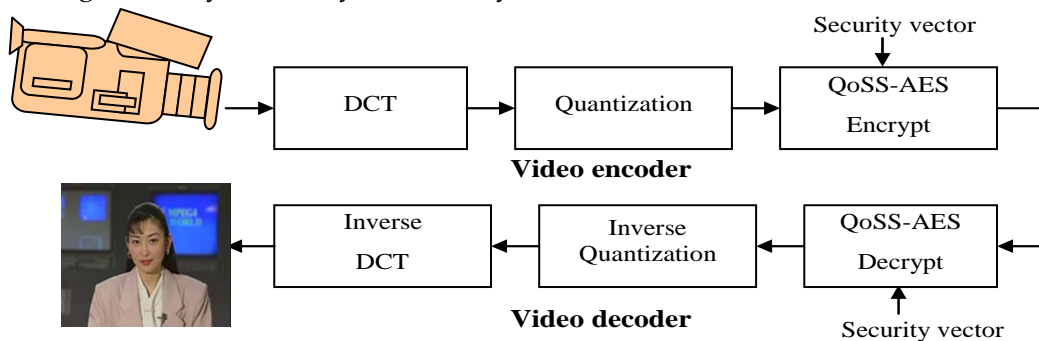


Figure 12. Secure video application Steps

Figure 12 shows the relationship between QoSs-AES processor and MPEG compression. After DCT transformation and quantization, the DC coefficients of

each 8x8 blocks are encrypted with the block of the QoSS-AES processor. On one hand, the DC coefficients are concatenated into a vector (128 bits), which is then encrypted by QoSS-AES processor. On the other hand, the QoSS-AES utilizes the security vector to perform the encryption process.

Other frequency algorithms for video encryption such as “Block Shuffle” [28] and “Subband Shuffle” [29] cannot provide the same level of security as QoSS-AES does. “Block Shuffle” and “Subband Shuffle” are strong to brute forth attack because DC coefficients are concealed by shuffling in a large data group.

To assess our proposed scheme through simulations, four standard video sequences “Carphone”, “Susie”, “Foreman”, “Salesman” (QCIF, one I-frame followed by 73 Pframes) have been used. The performance of our algorithm is compared to existing video encryption algorithms. In this experiment, QoSS-AES processor is applied to DC coefficients in all (I, P, B) - frames.

**Table 5. Effects of different algorithms on compression efficiency for one I-frame of “Susie” sequence**

Scramble Method	Size (bits)	Bit Overhead
Block Shuffle [28]	80312	100%
Subband Shuffle[29]	42888	9.1%
QoSS-AES (our)	39311	0%

As shown in Table 5, QoSS-AES generates no bit overhead to the MPEG bit-stream. On the contrary, “Subband Shuffle” and “Block Shuffle” generate 9 and 100% bit overhead to I-frame. Table 6 shows the processing time taken by our processor embedded in MPEG-4 video compression process.

The experimental results showed very low latency. This is due presumably to the high speed of QoSS-AES processor.

**Table 6. Processing overhead of QoSS-AES processor on four sequences**

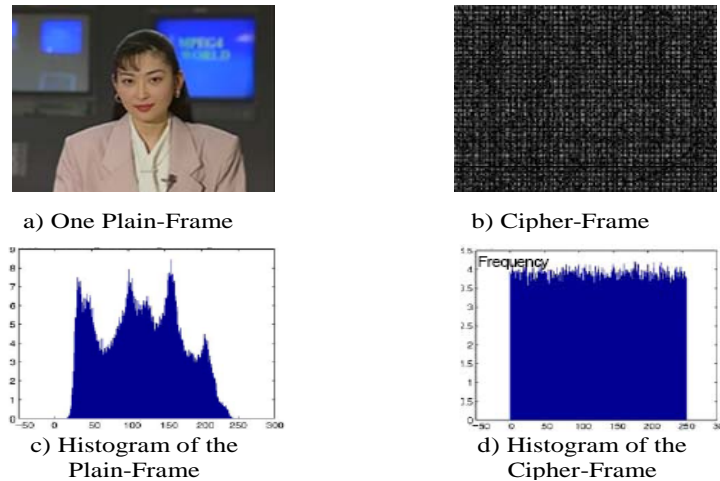
Video sequence	Original time (ms)	Bit numbers of all DC coefficients	QoSS-AES processor	
			Encryption time (ms)	Bit Overhead
“Carphone”	6782	1596	0.0024	0%
“Susie”	8407	1498	0.0023	0%
“Foreman”	7343	1816	0.0027	0%
“Salesaman”	4985	1555	0.0023	0%

Experimental results for different QoSS security vectors from 0000 to 0110 are reported in Table 7. For instance, the encryption with QoSS level 1 requires 0.00131 ms and 0.00164 ms in QoSS level 3. Additionally, we can see that when security level increases (from low to high) the encryption time increases as well but slightly. As a consequence, the encryption time goes 0,99  $\mu$ s from QoSS level 1 (AES-128 8 rounds) to QoSS level 12 (AES-256 14 rounds).

**Table7. Tradeoffs between encryption time and QoS requests on four sequences**

QoS	Security level	Encryption Time (ms)			
		Susie	Carphone	Foreman	Salesaman
0000	Low	0,00131	0,00139	0,00158	0,00136
0001	Low	0,00148	0,00157	0,00179	0,00153
0010	Medium	0,00164	0,00174	0,00198	0,0017
0011	Medium	0,00181	0,00192	0,00219	0,00188
0100	High	0,00197	0,0021	0,00238	0,00204
0101	High	0,00214	0,00228	0,00259	0,00222
0110	High	0,0023	0,00245	0,00278	0,00238

For an uncompressed digital video, we test the performance of the QoS-AES processor. In figure 13, we give the comparison of one plain I-frame of “akiyo” sequence and the cipher frame. It can be noticed that the plain-frame is encrypted to cipher-frame with uniform histogram, which implies the perfect cryptographic proprieties of QoS-AES processor.



**Figure 13. I-frame of encrypted “akiyo” sequence (one I-frame followed by 73 P-frames).**

Furthermore, QoS-AES processor can be used like hierarchical algorithms to provide a multilevel encryption system for digital MPEG video. Different combinations of security levels and computational overhead can be chosen to meet the requirements of various application scenarios. Therefore, QoS will configure the AES processor to switch from one security level to another during runtime, according to the need of the user.

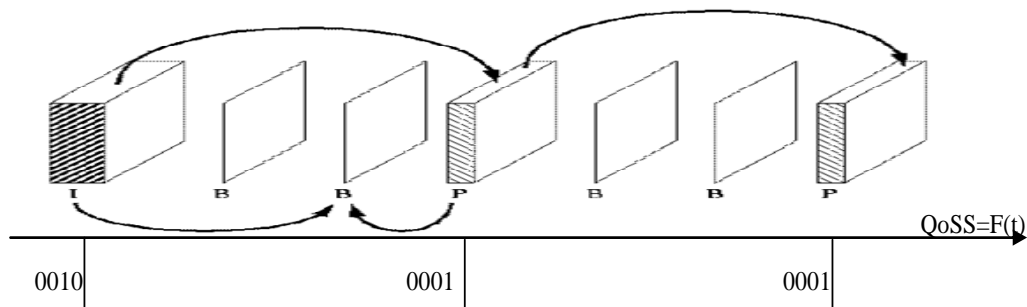
For example we can define four scenarios of QoS-AES MPEG encryption. As more scenarios are implemented, the security level is increased but so is the computational overhead. In the first scenario, the QoS-AES processor encrypts only the header information with QoS level one. In the second scenario, the processor encrypts I-frames with QoS level two in addition to the implementa-

tion in the first scenario. So, firstly QoSS-AES processor is configured to encrypt the header information, secondly the processor is arranged to ciphered I-frames. In the third scenario, I-frames are encrypted with QoSS level three and all of the I-blocks in the P-frames are encrypted with QoSS level two. In the fourth scenario, I-frames are encrypted with QoSS level four and all of the I-blocks in the P-frames and B-frames are encrypted with QoSS level two. Detailed information's among these scenarios are given in table 8.

**Table8. Defined Scenarios**

scenarios	Data				QoSS-AES				
	header information	I-Frame	I-blocks in the P-frames	I-blocks in the B-frames	Security level	QoSS			
						0	1	2	3
one	×				Low	×			
two	×	×			Medium	×	×		
three		×	×		High		×	×	
four		×	×	×	High		×		×

Figure 14 shows the QoSS-AES configuration in third scenario during MPEG video transmission.



**Figure 14. QoSS-AES Processor configuration process: Scenario three**

According to the table we can see the utility of the QoSS-Module; it's easy to switch from scenario to another one during runtime. Other hierarchical algorithms cannot provide the same simplicity and the same security level as QoSS-AES processor does.

## 6 Conclusion

Conventional cryptography has traditionally dealt with textual data. We extended the AES algorithm and applied it toward the cryptography of continuous video streams. This paper is aimed to investigate the quality of security ser-

---

vice for AES on embedded system for multimedia applications. To express the trade offs between security and real-time application requirements; we have proposed and developed a QoSS-AES processor by respecting two parameters: security and requirements. The performance of the proposed processor has been analyzed in an MPEG4 decoder. Experimental results revolved that the QoSS-AES processor offers an attractive alternative to conventional video encryption schemes by providing low communication latencies. The processor is specialized for MPEG's coding sequences and orders the level of protection provided in a hierarchy. Taking advantage of the inter-frame dependencies in MPEG, it may be possible to encrypt DC coefficients with QoSS level I, ( $I=1,2,\dots,12$ ). Enhanced protection is traded off against an increase of encryption time.

Future work will focus on improving and applying our QoSS module to more security variables such as authenticity and access control.

## References

- [1] National Institute of Standards and Technology (NIST). Data Encryption Standard (DES). Federal Information Processing Standards Publications (FIPS PUBS) 46-3, 1999.
- [2] E. Biham, and A. Shamir. Differential Cryptanalysis of DES-like Cryposystems. *Journal of Cryptology*, 4(1): 3–72, 1991.
- [3] National Institute of Standards and Technology (NIST). Advanced Encryption Standard (AES). Federal Information Processing Standards Publications (FIPS PUBS) 197-26, 2001.
- [4] H. Dobbertin, L. Knudsen, and M. Robshaw. The cryptanalysis of the AES – a brief survey. *Lecture Notes in Computer Science*, 3373: 1–10, 2005.
- [5] R. Phan. Impossible differential cryptanalysis of 7-round Advanced Encryption Standard (AES). *Information Processing Letters*, 91(1): 33–38, 2004.
- [6] S. Lucks. Attacking seven rounds of Rijndael under 192-bit and 256-bit keys. In *Proceedings of the Third Advanced Encryption Standard Candidate Conference*, pages 215–229, 2000.
- [7] H. Gilbert, and M. Minier. A collision attack on seven rounds of Rijndael. In *Proceedings of the Third Advanced Encryption Standard Candidate Conference*, pages 230– 241, 2000.
- [8] N. Ferguson, J. Kelsey, B. Schneier, M. Stay, D. Wagner, and D. Whiting. Improved cryptanalysis of Rijndael. In *Fast Software Encryption, 7th international Workshop, Lecture Notes in Computer Science*, 1978: 213–230, 2001.
- [9] D.D. Hwang, T. Kris, A. Hodjat, B.-C. Lai, S. Yang, P. Schaumont, and I. Verbauwhede. AES-Based Security Coprocessor IC in 0.18- $\mu$ m CMOS with

- 
- Resistance to Differential Power Analysis Side-Channel Attacks. *IEEE Journal of Solid-State Circuits*, 41(4): 781–792, 2006.
- [10] C.-H. Yen, and B.-F. Wu. Simple Error Detection Methods for Hardware Implementation of Advanced Encryption Standard. *IEEE Transactions on Computers*, 55(6): 720–731, 2006.
- [11] S. B. Ors, F. Gurkaynak, E. Oswald, and B. Preneel. Power-Analysis Attack on an ASIC AES implementation. In *Proceedings of the 2004 International Symposium on Information Technology Conference (ITCC 2004)*, pages 546–552, 2004.
- [12] I. Verbauwhede, P. Schaumont, and H. Kuo. Design and performance testing of a 2.29-Gb/s Rijndael processor. *IEEE Journal of Solid-State Circuits*, 38(3): 569–572, 2003.
- [13] C.-P. Su, T.-F. Lin, C.-T. Huang, and C.-W. Wu. A high-throughput low-cost AES processor. *IEEE Communications Magazine*, 41(12): 86–91, 2003.
- [14] C.-P. Su, C.-L. Horng, C.-T. Huang, and C.-W. Wu. A Configurable AES Processor for Enhanced Security. In *Proceedings of the 2005 conference on Asia South Pacific design automation ASP-DAC*, pages 361–366, 2005.
- [15] A. Hodjat, and I. Verbauwhede. Area-Throughput Trade-Offs for Fully Pipelined 30 to 70 Gbits/s AES Processors. *IEEE Transactions on Computers*, 55(4): 366–372, 2006.
- [16] D. Shang, F. Burns, A. Bystrov, A. Koelmans, D. Sokolov, and A. Yakovlev. High-security asynchronous circuit implementation of AES. *IEEE Proceedings Computers and Digital Techniques*, 153(2):71–77, 2006.
- [17] N. Sklavos, and O. Koufopavlou. Architectures and VLSI implementations of the AES-proposal Rijndael. *IEEE Transactions on Computers*, 51(12): 1454–1459, 2002.
- [18] C. Chitu, D. Chien, C. Chien, I. Verbauwhede, and F. Chang. A hardware implementation in FPGA of the Rijndael algorithm. In *Circuits and Systems (MWSCAS-2002) 45th Midwest Symposium*, pages 507–510, 2002.
- [19] S. Mangard, M. Aigner, and S. Dominikus. A highly regular and scalable AES hardware architecture. *IEEE Transactions on Computers*, 52(4): 483–491, 2003.
- [20] N.-A. Saqib, F. Rodriguez-Henriquez, and A. Diaz-Perez. AES algorithm implementation—an efficient approach for sequential and pipeline architectures. In *Proceedings of the Fourth Mexican International Conference on Computer Science*, pages 126–130, 2003.
- [21] A. Brokalakis, A.P. Kakarountas, and C.E. Goutis. A High-Throughput Area Efficient FPGA Implementation of AES-128 Encryption. *IEEE Workshop on Signal Processing Systems Design and Implementation (SIPS)*, pages 116–121, 2005.
-

- 
- [22] N. Nedjah, L. de Macedo Mourelle, and M.P Cardoso. A Compact Piplined Hardware Implementation of the AES-128 Cipher. Proceedings of the Third International Conference on Information Technology: New Generations, pages 216–221, 2006.
- [23] F. Standaert, G. Rouvroy, J.-J. Quisquater, and J.-D. Legat. A Methodology to Implement Block Ciphers in Reconfigurable Hardware and its Application to Fast and Compact AES RIJNDAEL. Proceedings of the eleventh international symposium on Field programmable gate arrays, ACM/SIGDA, pages 216–224, 2003.
- [24] A.-B. Ignacio, F.-U. Claudia, and R. Cumplido. Design and Implementation of an FPGA-Based 1.452-Gbps Non-pipelined AES Architecture. Lectures Notes in Computer Science, 3982: 446–455, 2006.
- [25] C. Aurrecochea, A. Campbell, and L. Hauw. A Survey of Quality of Service Architectures. Multimedia Systems Journal, 6(3):138-151, 1998.
- [26] L. R. Welch, B. A. Shirazi, B. Ravindran, and C.Bruggeman. DeSiDeRaTa: QoS Management Technology for Dynamic, Scalable, Dependable, Real-Time Systems. In Proceedings of the 15th Symposium on Distributed Computer Control Systems (DCCS98), IFAC,1998.
- [27] Z. Medien, M. Mohsen, K. Lazhar, B. Adel, and T. Rached. A Modified AES Based Algorithm for Image Encryption. Int. Journal of Computer Science and Engineering. 1(2):70–75, 2007.
- [28] L. Tang. Methods for encrypting and decrypting MPEG video data efficiently. In Proceedings of the fourth ACM international conference on Multimedia, pages 219–229, 1996.
- [29] W. Zeng, and S. Lei. Efficient frequency domain selective scrambling of digital video. IEEE Transactions on Multimedia, 5(1): 118–129, 2003.