Convolutional neural network based key generation for security of data through encryption with advanced encryption standard

Ismail Negabi, Smail Ait El Asri, Samir El Adib, Naoufal Raissouni
Remote Sensing and GIS Unit, National School of Applied Sciences, University of Abdelmalek Essaadi, Tetuan, Morocco

Article Info

Article history:

Received Apr 4, 2022
Revised Aug 5, 2022
Accepted Sep 2, 2022

Keywords:

Advanced encryption standard
Convolutional neural networks
Cryptography
Deep learning
Embedded systems
Field programmable gate arrays
Hardware description language

ABSTRACT

Machine learning techniques, especially deep learning, are playing an increasingly important role in our lives. Deep learning uses different models to extract information from the data. They have already had a huge impact in areas such as health (i.e., cancer diagnosis), self-driving cars, speech recognition, and data encryption. Recently, deep learning models, including convolutional neural networks (CNN), have been proven to be more effective in the security field. Moreover, the National Institute of Standards and Technology (NIST) recommends the advanced encryption standard (AES) algorithm as the most often utilized encryption method in several security applications. In this paper, a crypt-intelligent system (CIS) capable of securing data is proposed. It is based on the combination of the performance of CNN with the AES, by substituting the key expansion unit of AES with a CNN architecture that performs the key generation. Our CIS is described using very high-speed integrated circuit (VHSIC) hardware description language (VHDL), simulated by ModelSim, synthesized, and implemented with Xilinx ISE 14.7. Finally, the Airtex-7 series XC7A100T device has achieved an encryption throughput of 965.88 Mbps. In addition, the CIS offers a high degree of flexibility and is supported by reconfigurability, based on the experimental results, if sufficient resources are available, the architecture can provide performance that can satisfy cryptographic applications.

Corresponding Author:

Ismail Negabi
Remote Sensing and Geographic Information System Unit, National School of Applied Sciences, University of Abdelmalek Essaadi
Tetuan, Morocco
Email: ismail.negabi@etu.uae.ac.ma

1. INTRODUCTION

In the context of deep learning models [1]–[4], a convolutional neural network (CNN) is a type of intelligent system intended to imitate the biological neurons network found in the human mind [2], [5]. CNN is used to classify data, detect objects, encryption and decryption data, and generate encryption keys [6]. Generally, CNN architecture is composed of a number of convolutional layers, pooling layers, and fully connected layers [1], [7], [8]. CNNs are linear layers that share their weights in space and are able to recognize patterns at different locations. The pooling layers are non-linear, reducing the size of the space to minimize the number of parameters. The most popular pooling functions are max pooling [4], [8], which returns the highest value in a specified range, average pooling, which returns the average value for a given range, and fully connected layers which return results that depend on the entire input.

Data security [9] is becoming crucial for a variety of embedded applications. Resilience to attacks on cryptographic systems is one of the main properties that cryptographic algorithms have to offer. In many security applications, advanced encryption standard (AES) [10]–[12] has surpassed data encryption standard (DES) [13]–[15] as the most extensively utilized and secure encryption algorithm. It provides the perfect integration of security and different encryption key sizes, efficiency, performance, ease of implementation and flexibility [11], [12], [16], although the size of the key influences security strength. The problem of energy consumption has lately become more prominent, particularly in embedded systems used in digital devices, where smaller space and reduced energy consumption are critical. The AES algorithm requires extensive hardware/software implementation. In embedded system applications, hardware implementation is the most expensive component [10]–[12], [16].

In this article, a CNN architecture is proposed to generate encryption keys used by the AES [12], [17] algorithm to build smart cryptosystems capable of protecting data. This technique is simpler to implement than using standard AES and non-linear operations. CNN [7] should be reversible and capable of encrypting and decrypting plaintext/ciphertext with very small error rates. This includes considering the architectural design of the CNN, modifying it in software, and testing the simulation results to provide an AES-128 based CNN. The internal structure of CNN components will be considered in the design, and further research will be conducted on issues that have a significant impact on the implementation of CNNs.

2. BACKGROUND

CNNs are a subset of artificial neural networks (ANN) that use convolution rather than general matrix multiplication in at least one of their layers. The combination of CNN performance and the AES algorithm improves data security. This section provides detailed information on CNN and the AES algorithm.

2.1. Convolutional neural networks

Deep learning performs best than other network learning algorithms [1], [2], [4], [18], as it can obtain more abstract high-level features from the raw input. Today, deep learning is a class of machine learning [7], [9] algorithms with different variants of deep architecture. CNN and stacked auto-encoders (SAE) are the most prevalent and popular architectures [2], [7]. However, they are extremely slow to train, especially when dealing with huge databases. CNNs are deep neural networks (DNNs) and biologically inspired variants of multilayer perceptron [7]. It is often used in images/videos, computer vision, and data encryption/decryption. We distinguish two parts, the first part, which we call the convolutional part of the model, and the second part, which we will call the classification part of the model corresponding to a multilayer perceptron model. The general architecture of CNN is shown in Figure 1.

– Convolution layer (Conv): This layer is essential to CNN and is always present as at least the first layer. Its goal is to identify the existence of a certain set of characteristics in the input data. Its data are convolved with some filters, while the convolution operators' function which is based on three hyper-parameters (depth, pitch, and margin) allows sizing of the volume of the convolution layer.
– Max pooling layer (Pool): Frequently, the pooling layer is inserted between two convolution layers: it takes a set of feature maps as input and performs the pooling operation. The purpose of this process is to reduce the size of the data while retaining their vital properties. Using the max operation, this layer operates independently on each depth slice of the input data and resizes it spatially.
- Fully connected layer (FC): This is the final layer of a CNN. It performs a weighted sum and potentially an activation function to the values received as input, which is a vector and generates a new vector as output.

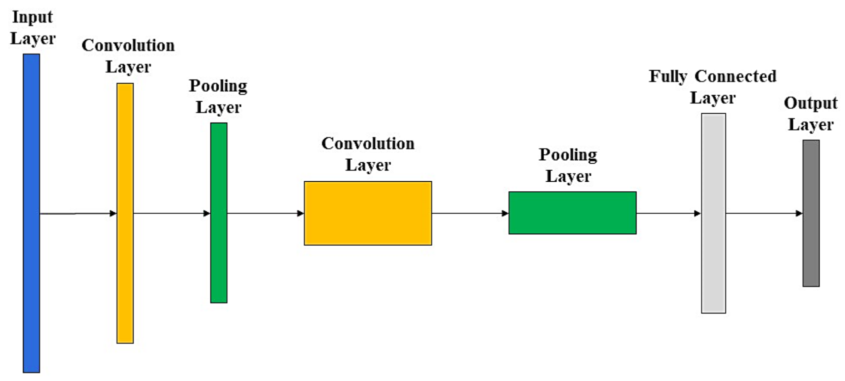

Figure 1. Basic CNN architecture

2.2. Advanced encryption standard (AES)

The advanced encryption standard (AES) is a block cipher that employs a 4×4 array with input, output and state block 128-bit [3], [7], [10], [19]. This is identified by $\mathrm{N_b} = 4$ which illustrates the number of words of 32-bits in the state matrix. The length of the encryption key on the AES algorithm is 128, 192, or 256-bit, these lengths are identified by $\mathrm{N_k}$ [11], [12], [16]. It determines the number of rounds to execute when running the AES, this number is identified by $\mathrm{N_r}$, where $\mathrm{N_r} = 10$ when $\mathrm{N_k} = 4$, $\mathrm{N_r} = 12$ when $\mathrm{N_k} = 6$, and $\mathrm{N_r} = 14$ when $\mathrm{N_k} = 8$ [20]. Table 1 shows the AES parameters (key, block, and round) that meet this requirement.

Table 1. AES parameters

Algorithm AES	Key Length (N_k words)	Block Size (N_b words)	No of Rounds (N_r)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

The AES encryption algorithm performs a round function consisting of 4 operations for encryption and decryption as shown in Figure 2: i) the *SubBytes* transformation, which is based on the S-box table; ii) the *ShiftRows* transformation; iii) the *MixColumns* transformation, which shuffles data in each column of the state matrix, and iv) the *AddRoundKey* transformation, which adds a round key to the state matrix [10], [19], [20].

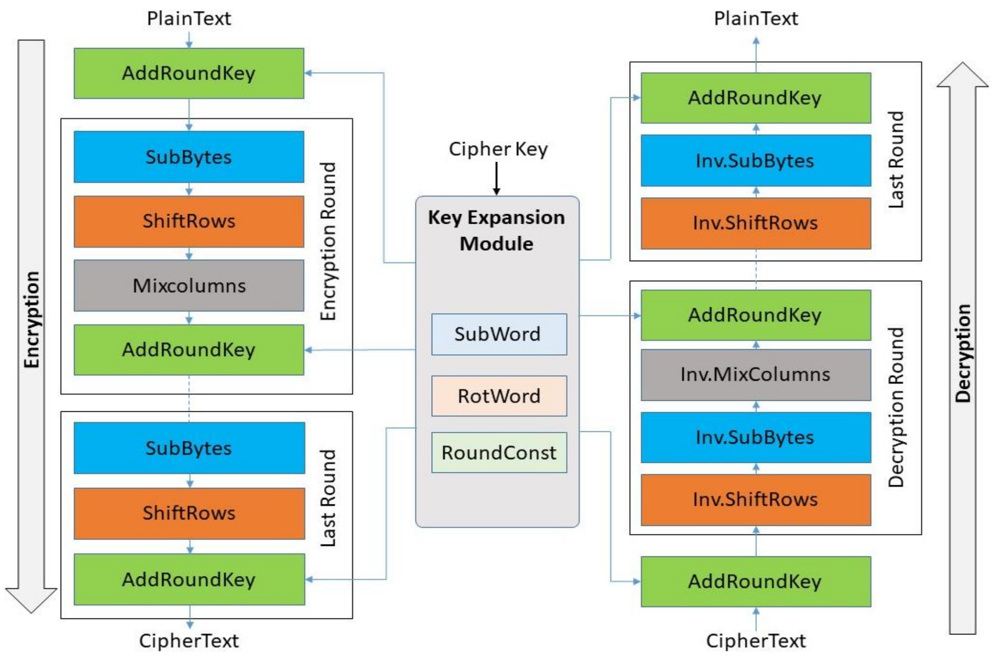

Figure 2. Block diagram for AES encryption and decryption

2.2.1. Encryption

The input data is sent to the "State" matrix for encryption. After the first round of encryption key addition, the state matrix is modified, using 10, 12, or 14 rounding functions (depending upon the key size), with the last round notably deviating from the first Nr-1 rounds. After that, the last state is transmitted to the output data. The round function is configured with a key expansion schedule, which consists of a 1-D vector of 4-byte words. The transformations of the AES algorithm are *SubBytes* (SB), *ShiftRows* (SR), *MixColumns* (MC), and *AddRoundKey* (AK) [10], [16], [19], [21].

$$State = \begin{bmatrix} d_{15} & d_{11} & d_7 & d_3 \\ d_{14} & d_{10} & d_6 & d_2 \\ d_{13} & d_9 & d_5 & d_1 \\ d_{12} & d_8 & d_4 & d_0 \end{bmatrix} \tag{1}$$

a. *SubBytes* (SB) transformation

The SB transformation is a nonlinear byte change that uses a substitution table S-box to work separately on each byte of the state using the S-box table. The values these 256 integers (0 to 255) create are listed in this table. Thus, the SB transformation on the state is described by (2).

$$SB(State) = \begin{bmatrix} SB(d_{15}) & SB(d_{11}) & SB(d_7) & SB(d_3) \\ SB(d_{14}) & SB(d_{10}) & SB(d_6) & SB(d_2) \\ SB(d_{13}) & SB(d_9) & SB(d_5) & SB(d_1) \\ SB(d_{12}) & SB(d_8) & SB(d_4) & SB(d_0) \end{bmatrix} \quad (2)$$

b. *ShiftRows* (SR) transformation

The bytes in the final three rows of the report are shifted to the left during SR transformation by varying amounts of bytes. The first row (r = 0) is unshifted, the second row (r=1) is shifted by one byte, the third row (r=2) by two bytes, and the last row (r=3) by three bytes. As a result, the SR transformation is described by (3).

$$Z = SR(SB(State)) = \begin{bmatrix} SB(d_{15}) & SB(d_{11}) & SB(d_7) & SB(d_3) \\ SB(d_{10}) & SB(d_6) & SB(d_2) & SB(d_{14}) \\ SB(d_5) & SB(d_1) & SB(d_{13}) & SB(d_9) \\ SB(d_0) & SB(d_{12}) & SB(d_8) & SB(d_4) \end{bmatrix} \quad (3)$$

c. *MixColumns* (MC) transformation

Each column of the State is treated as a four-term polynomial by the MC transformation, which acts column-by-column. Considered to be polynomials over $GF(2^8)$, the columns are multiplied modulo x^4+1 by a constant polynomial $a(x)$, which is (4).

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} \quad (4)$$

This may be represented by the matrix multiplication function (5) and (6).

$$R(x) = a(x) \otimes Z(x) \quad (5)$$

$$\begin{bmatrix} R_{0,c} \\ R_{1,c} \\ R_{2,c} \\ R_{3,c} \end{bmatrix} = MC\left(SR\left(SB\ (State)\right)\right) = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \otimes \begin{bmatrix} Z_{0,c} \\ Z_{1,c} \\ Z_{2,c} \\ Z_{3,c} \end{bmatrix} for\ 0 \le c \le N_b \quad (6)$$

This multiplication matrix replaces the 4-bytes in each column with:

$$R_0 = (\{02\}.Z_0) \oplus (\{03\}.Z_1) \oplus Z_2 \oplus Z_3$$
$$R_1 = Z_0 \oplus (\{02\}.Z_1) \oplus (\{03\}.Z_2) \oplus Z_3$$
$$R_2 = Z_0 \oplus Z_1 \oplus (\{02\}.Z_2) \oplus (\{03\}.Z_3)$$
$$R_3 = (\{03\}.Z_0) \oplus Z_1 \oplus Z_2 \oplus (\{02\}.Z_3)$$

d. *AddRoundKey* (AK) transformation

During *AddRoundKey* (AK), each bit of the state is added with the round key (rk) using the xor function. Each round key is composed of N_b words from the key schedule. Each of these N_b words are added to the state's columns, resulting in (7).

$$AK(R) = \begin{bmatrix} R_{15} & R_{11} & R_7 & R_3 \\ R_{14} & R_{10} & R_6 & R_2 \\ R_{13} & R_9 & R_5 & R_1 \\ R_{12} & R_8 & R_4 & R_0 \end{bmatrix} \oplus \begin{bmatrix} rk_{15} & rk_{11} & rk_7 & rk_3 \\ rk_{14} & rk_{10} & rk_6 & rk_2 \\ rk_{13} & rk_9 & rk_5 & rk_1 \\ rk_{12} & rk_8 & rk_4 & rk_0 \end{bmatrix} \quad (7)$$

In the AES-128 algorithm, the first (rk^0), which is the initial encryption key, is used in the additional AK at the beginning of the first round. For each AK transformation, the round key (rk) is calculated with the key schedule. rk^i, where $1 \le i \le 10$, is determined from the last rk^{i-1}. Let $q(j)$ $(0 \le j \le 3)$ be the column j of rk^{i-1} and let $w(j)$ be the column j of rk^i. Then the updated rk^i is determined as:

$$\mathrm{w}(0) = \mathrm{q}(0) \oplus \left(\mathrm{Rot}\left(\mathrm{SB}\left(\mathrm{q}(3)\right)\right)\right) \oplus \mathrm{rcon}^{\mathrm{i}}$$
$$\mathrm{w}(1) = \mathrm{q}(1) \oplus \mathrm{w}(0)$$
$$\mathrm{w}(2) = \mathrm{q}(2) \oplus \mathrm{w}(1)$$
$$\mathrm{w}(3) = \mathrm{q}(3) \oplus \mathrm{w}(2)$$

The $\mathrm{rcon}^{\mathrm{i}}$ (round constant) contain values $[02^{i-1}; \{00\}; \{00\}; \{00\}]$. Rot is a function that shifts over 1-byte from a 4-byte input.

2.2.2. Key expansion

The AES encryption algorithm uses an encryption key and a key expansion to build a key schedule [20]. It produced a total of $\mathrm{N_b}$ $(\mathrm{N_r}+1)$ words: a first set of $\mathrm{N_b}$ words are needed by the algorithm, and every round $\mathrm{N_r}$ needs $\mathrm{N_b}$ words of key data. The final key schedule $[w_i]$ is a linear array of 4-byte words, with i being a value between 0 and $\mathrm{N_b}(\mathrm{N_r}+1)$.

SubWord, which is a function, takes four bytes as the input words and generates output words by applying the S-box table to each of them. The *RotWord* takes as input words $[a_1, a_2, a_3, a_4]$, executes a cyclic shift, and returns the word $[a_1, a_2, a_3, a_4]$. The values of the round constant word array, $\mathrm{Rcon}[i]$ has the values given by $[x^{i-1}, \{00\}, \{00\}, \{00\}]$, with x^{i-1} are powers of x (denoted as $\{02\}$) in the $\mathrm{GF}(2^8)$ field.

In order to fill the encryption Key, the expanded key's first $\mathrm{N_k}$ words are used. Every word after that, w[i], is the xor function of the word before it, $w[i-1]$ and $\mathrm{N_k}$ place earlier $w\,[i-\mathrm{N_k}]$. For words in place that are a multiple of $\mathrm{N_k}$, a process is done to $w\,[i-1]$ before to the xor function, followed by a xor with a $\mathrm{Rcon}[i]$. This process is a cyclic rotation of the bytes in a *RotWord*, followed by a table search on the 4-bytes of the *SubWord*. It is significant to mentioned that the key expansion process for 256-bit encryption keys $(\mathrm{N_k} = 8)$ is somewhat various from that for 128 and 192-bit. The *SubWord* is performed to $w[i-1]$ before the xor function if $\mathrm{N_k} = 8$ and $i-4$ is a multiple of $\mathrm{N_k}$ as shown in Algorithm 1.

Algorithm 1. Key expansion
```
Key expansion (byte key[4*Nk], word w[Nb*(Nr+1)]Nk
begin
   word temp
   i=0
   while (i<Nk)
      w[i]=word(key[4*i+1], key[4*i+2], key[4*i+3])
      i=i+1
   end while

   i=Nk
   while(i<Nb*(Nr+1))
      temp=w[i-1]
      if(i mod Nk=0)
         temp=subword(rotword(temp)) xor rcon[i/Nk]
      else if(Nk>6 and i mod Nk=4)
         temp=subword (temp)
      end if
         w[i]=w[i-Nk] xor temp
         i = i+1
   end while
end
```

2.2.3. Decryption

The reverse encryption transformations (*InvSubBytes* (Inv SB), *InvShiftRows* (Inv SR), *InvMixColumns* (Inv MC), and *AddRoundKey* (AK)) for the AES encryption algorithm are built in the opposite direction of the encryption process. Each encryption transformation step is carried out in the opposite direction as shown in Figure 2. The main reverse encryption rounds are likewise performed in the opposite direction. AK is followed by a byte substitution, a reverse row shift, a reverse column shuffle, and a key addition round. The regulations for the final round remain unchanged. The reverse column shuffle step is similar to the encryption column shuffle step, but it uses a different fixed polynomial. The bytes in the row are cycled by an appropriate offset in the row. The first row has not been altered in any manner. The second and third rows have been shifted to the right [16], [20].

3. RELATED WORK

In the past decades, research on combining deep learning models with cryptographic algorithms has gained importance. Many academic researchers have proposed various methods for designing intelligent systems capable of encrypting and decrypting data. The majority of these architectures are also efficient in terms of execution time, security level, and propagation latency.

Jogdand and Bisalapur [22] suggested a key generation technique based on neural networks. This technique's performance has been assessed in terms of security, synchronization time, and unpredictability. While performing key generation, the current work also performs randomization. They recommended that in the future, the generated keys should be distributed securely. For encryption and decryption, Satapathy *et al*. [23] described a neural key generation using a Hopfield network (HNBNKG). Instead of providing the secret key, their solution assures that the weight vectors of the trained network may be exchanged between the sender and receiver to prevent attacks. This has also been prioritized in the work given.

The AES encryption algorithm can manage any plaintext and key size combination (128, 192, or 256-bit). Many researchers have used various approaches for implementation based on various technical criteria, such as AES strength execution, AES for executing with efficiency or effectiveness, and AES for implementing hardware and software. In the literature, several hardware solutions have been published, some of them make use of field programmable gate arrays (FPGAs), while others make use of application-specific integrated circuits (ASICs) [17]. Adib and Raissouni [16] explain that Rijmen suggested an AES hardware implementation based on composite fields, which was the first important step in compacting the AES implementation. Rijmen also presented a similar approach concept that has been used in FPGA and ASIC. Unfortunately, the majority of those solutions are too expensive for use in real-world embedded applications.

4. PROPOSED CRYPT-INTELLIGENT SYSTEM

In this section, the main idea is to design a crypt-intelligent system (CIS) based on CNN and AES-128, which can perform encryption and decryption processes with high performance and extremely low error rates. CNNs are a powerful tool for controlling nonlinear systems. Because the network's purpose is to limit the system's cracking ability by utilizing non-linear activation functions, its non-linear approximation characteristic makes it more effective in practical applications. Our proposed cryptosystem is based on the AES algorithm, a CNN network, a memory unit module, and a control unit module as shown in Figure 3. These units are presented in the following subsections.

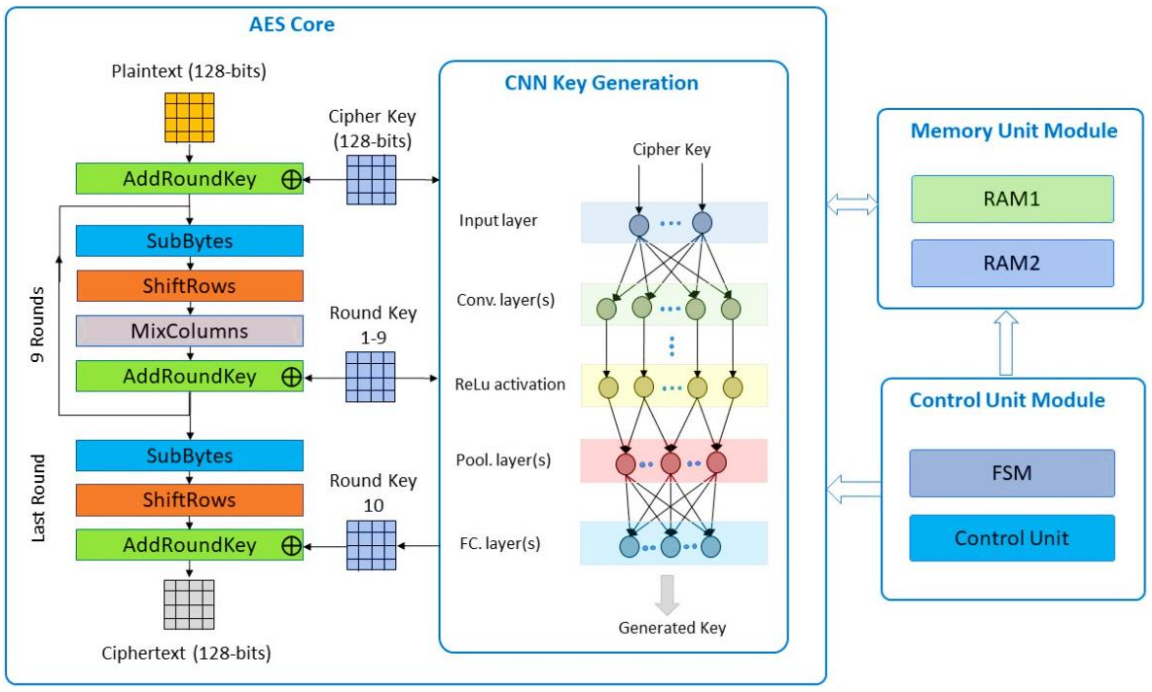

Figure 3. The proposed CIS for AES-128 implementation with CNN

4.1. CNN execution on CIS

The authors proposed the use of CNN for key generation, which has included the performance of CNN in the encryption/decryption process. The combination of CNN with AES is achieved by substituting the key expansion module in the AES with CNN architecture capable of generating the keys similar to the substituted module with the introduction of nonlinearity of CNNs. The CNN module takes as input a matrix of size 4×32 that corresponds to the encryption key divided into four blocks of size 32-bits. The output is a matrix of size 10×128, with each row representing a generated key that will be used in the steps of the AES algorithm. This module generates 10 keys at a time and stores them in the RAM, the control unit is responsible for using them at specific moments in the system operation.

In order to train the CNN module, a script programmed in Python language is used to generate a dataset that consists of the input keys and those generated based on the operation of the key expansion module of the AES. The module was trained on the generated dataset using "Adam" as the weight optimization algorithm. The activation function utilized in the output layer is Softmax, whereas the activation function used in the other layers is rectified linear unit (ReLU). Binary-cross entropy was used as the error function. The 128-bit input keys are randomly generated, which ensures that most possible cases for an encryption key are covered. The generated dataset is divided into two parts: 80% for training the module and 20% for validation.

4.2. AES execution on CIS

The AES performs 4 transformations (*SubBytes* (SB), *ShiftRows* (SR), *MixColumns* (MC), and *AddRoundKey* (AK)) on each round. The description below explains how the lookup tables and corresponding operations of the AES round are obtained: The first quarter of the round's SubBytes (SB) and *MixColumns* (MC) operations can be described as (8) and (9).

$$H = MC\left(SB\left(SR(State)\right)\right) = A(x) \otimes SB\left(SR(State)\right) \tag{8}$$

$$A(x) = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \tag{9}$$

The state matrix is the transformed data, H is the result of the transformation of MC, where A(x) is the multiplicative vectors matrix. It is possible to use a logarithm table to do the aforementioned multiplication. The implementation of the AES is based on combining the MC and SB transformations in one convenient table. Compared to the 8×32 bit wide T-box tables, the 4 tables (G_0 to G_3) used are 8×8 bit, containing 256 values (0 to 255) given as [16]:

$$G_0(a) = Log'\left(\left(Log(01)\right) + \left(Log\left(SB(a)\right)\right)\right)$$
$$G_1(a) = Log'\left(\left(Log(01)\right) + \left(Log\left(SB(a)\right)\right)\right)$$
$$G_2(a) = Log'\left(\left(Log(02)\right) + \left(Log\left(SB(a)\right)\right)\right)$$
$$G_3(a) = Log'\left(\left(Log(03)\right) + \left(Log\left(SB(a)\right)\right)\right)$$

Such that a represents the elements of the S-box table, SB represents the transformation $SubBytes(SB)$ and Log/Log' represents Logarithm/Antilogarithm respectively. The results from 4 tables (G_0 to G_3) will be XORed to produce the final result (H), as shown by:

$$H_{15} = G_2(d_{15})\ xor\ G_3(d_{10})\ xor\ G_1(d_5)\ xor\ G_0(d_0)\ xor\ rk_{15}$$
$$H_{14} = G_0(d_{15})\ xor\ G_2(d_{10})\ xor\ G_3(d_5)\ xor\ G_1(d_0)\ xor\ rk_{14}$$
$$H_{13} = G_1(d_{15})\ xor\ G_0(d_{10})\ xor\ G_2(d_5)\ xor\ G_3(d_0)\ xor\ rk_{13}$$
$$H_{12} = G_3(d_{15})\ xor\ G_1(d_{10})\ xor\ G_0(d_5)\ xor\ G_2(d_0)\ xor\ rk_{12}$$
$$H_{11} = G_2(d_{11})\ xor\ G_3(d_6)\ xor\ G_1(d_1)\ xor\ G_0(d_{12})\ xor\ rk_{11}$$
$$H_{10} = G_0(d_{11})\ xor\ G_2(d_6)\ xor\ G_3(d_1)\ xor\ G_1(d_{12})\ xor\ rk_{10}$$
$$H_9 = G_1(d_{11})\ xor\ G_0(d_6)\ xor\ G_2(d_1)\ xor\ G_3(d_{12})\ xor\ rk_9$$
$$H_8 = G_3(d_{11})\ xor\ G_1(d_6)\ xor\ G_0(d_1)\ xor\ G_2(d_{12})\ xor\ rk_8$$
$$H_7 = G_2(d_7)\ xor\ G_3(d_2)\ xor\ G_1(d_{13})\ xor\ G_0(d_8)\ xor\ rk_7$$
$$H_6 = G_0(d_7)\ xor\ G_2(d_2)\ xor\ G_3(d_{13})\ xor\ G_1(d_8)\ xor\ rk_6$$
$$H_5 = G_1(d_7)\ xor\ G_0(d_2)\ xor\ G_2(d_{13})\ xor\ G_3(d_8)\ xor\ rk_5$$
$$H_4 = G_3(d_7)\ xor\ G_1(d_2)\ xor\ G_0(d_{13})\ xor\ G_2(d_8)\ xor\ rk_4$$
$$H_3 = G_2(d_3)\ xor\ G_3(d_{14})\ xor\ G_1(d_9)\ xor\ G_0(d_4)\ xor\ rk_3$$

$$H_2 = G_0(d_3)\ xor\ G_2(d_{14})\ xor\ G_3(d_9)\ xor\ G_1(d_4)\ xor\ rk_2$$
$$H_1 = G_1(d_3)\ xor\ G_0(d_{14})\ xor\ G_2(d_9)\ xor\ G_3(d_4)\ xor\ rk_1$$
$$H_0 = G_3(d_3)\ xor\ G_1(d_{14})\ xor\ G_0(d_9)\ xor\ G_2(d_4)\ xor\ rk_0$$

4.3. Memory unit module

The memory unit consists of two RAMs that operate in parallel. The memory may hold data as well as 32-bit instruction words. Encrypting or decrypting data, input/output data, and any data necessary for the encryption and decryption process. Therefore, advanced encryption standard (AES): S-box, logarithm tables, and round keys (rk) which is generated by the CNN module, are saved in RAM0 and RAM1, respectively. Each RAM is 512×8-bit dual ported for faster performance and access times.

4.4. Control unit module

Our proposed system is primarily built around a control unit module and a finite state machine (FSM). Its primary role is to control other units based on instructions saved in the RAMs. The FSM is responsible for controlling the processor of our CIS system in addition to producing status/control signals during data transfer, encryption, and embedded block memories (BRAM) addresses for accessing and modifying data in the memory. Through several multiplexers that were integrated into the architecture, the controller is intelligent enough to conduct transformations on various iterations.

5. RESULTS AND DISCUSSION

Our proposed CIS was coded in very high-speed integrated circuit (VHSIC) hardware description language (VHDL) and synthesized with Xilinx ISE 14.7 using the ModelSim simulator. The target device selected was the Artix-7 (XC7A100T). According to the synthesis findings, 999 slices are utilized by the synthetic device. The device operates at a clock speed of 377.301 MHz using just 2-BRAM 512×8 bits Dual-Port Wide. The results obtained are well organized in the form of tables' forms starting from Tables 2-7. Table 2 lists the platform characteristics of our proposed CIS implementation (hardware and software configurations). Table 3 summarizes the FPGA specifications utilized in the construction of our proposed CIS. Tables 4 and 5 show the details of the device used for the implementation of AES-128 and the CNN key generation. Table 6 shows the details of the timing summary.

Table 2. Implementation platform specifications

Implementation	
Hardware	Specification
Core processor	Intel(R) Core i5@2.4 GHz
Operating system	Windows 10 pro/Xilinx ISE 14.7 (64-bit)
RAM	8 GB

Table 3. FPGA characteristics

Parameters	Values
Family	Artix-7
Device	XC7A100T
Package	CSG324
Speed Grade	-3

Table 4. Device utilization summary for AES-128 encryption

Slice Logic Utilization			
Parameters	Used	Available	Utilization
No. of Slice Registers	787	35200	2%
No. of Slice LUT's	873	17600	4%
No. of Fully used LUT-FF Pairs	524	1136	46%

Table 5. Device utilization summary for CNN key generation

Slice Logic Utilization			
Parameters	Used	Available	Utilization
No. of Slice Registers	212	12480	1%
No. of Slice LUT's	211	12480	1%
No. of Fully used LUT-FF Pairs	138	285	48%

Table 6. Timing summary for our implementation CIS

Parameters	Values
Minimum Period	2.650 ns
Minimum Input arrival time before clock	0.976 ns
Maximum Output required time after clock	1.247 ns
Maximum Frequency	377.301 MHz
Maximum Combinational Path Delay	0.670 ns

In order to ensure a superior result in terms of both area and throughput, the proposed system is implemented by Xilinx ISE 14.7 and FPGA device Artix-7 (XC7A100T) used for downloading. Table 7 displays a summary of the device use of the whole algorithm on the same hardware. Figure 4 shows the simulation result of our proposed system (CIS). It takes as input a plaintext of size 128-bits and an encryption key of size 128-bits. Its output is a ciphertext of size 128-bits.

Table 7. Implementation results of our CIS using FPGA devices of the Xilinx Artix-7

Parameters	Values
Target FPGA device	Artix-7 XC7A100T
Maximum Frequency	377.301 MHz
No. of Slice Registers	999
No. of Fully used LUT-FF Pairs	662
No. of Slice LUT's	1084
Block RAMs	2

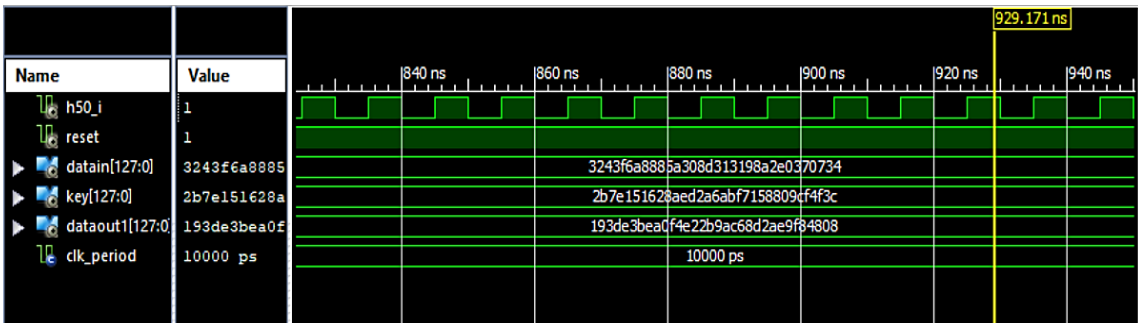

Figure 4. Simulation result of proposed CIS

A comparison is made between the proposed cryptosystem architecture and multiple FPGA device implementations of diverse designs. The proposed device is built in Artix-7 in order to provide sufficient memory to perform AES-128 Rijndael with a CNN key generator. For input recording, each round is required in one clock cycle, so the total clock cycle required to process 128-bit data for AES-128 is 12 clocks. Table 8 displays the throughput determined by several studies. The throughput and efficiency are calculated manually, defined as (10) and (11).

$$Throughput = \frac{\#\ of\ bits \times frequency}{\#\ of\ Clock\ Cycles} \tag{10}$$

$$Efficiency = \frac{Throughput}{\#\ of\ Used\ Slices} \tag{11}$$

Table 8. Comparison of the CIS with other FPGA implementations

Design	Year	Platform Device	Mode (Enc/Dec)	Max. Freq. [MHz]	Throughput [Mbps]	Area (Slices)	Efficiency [Mbps]/Slices
[24]	2016	Virtex-7 XC7VX690T	Enc	208.07	1280	3760	0.34
[25]	2018	Artix-7 XC7A100T	Enc	291.63	888.80	989	0.90
[26]	2020	Artix-7 XC7A100T	Enc	100	1792	4568	0.39
This Work (without CNN)	2022	Artix-7 XC7A100T	Enc	282.65	347.87	588	0.59
This Work (with CNN)	2022	Artix-7 XC7A100T	Enc	377.30	965.88	999	0.96

6. CONCLUSION

Deep learning models and the AES algorithm were used to design our proposed CIS. It has been enhanced and updated for data encryption. The selection of these algorithms (AES and CNN) is accurate and efficient and has been widely recognized by the scientific community. Our CIS has a higher throughput than other cryptographic processors based on deep learning methods and the AES encryption algorithm. Even when compared to some recent architecture, the implementation results are highly promising. The maximum frequency achieved is 377.30 MHz, which is comparable to other designs presented in the literature. Furthermore, our CIS provides a great level of flexibility and reconfigurability. According to the experimental results obtained, if sufficient resources are available, the architecture can provide performance that can satisfy cryptographic applications. Much better performance can be achieved if the network is implemented as an integrated circuit chip instead of being prototyped on FPGA.

In the future, our CIS will be able to use different key sizes (128, 192, or 256 bits) for the encryption or decryption of data (text, image). We will also add a detection unit for the different attacks to which our CIS can be exposed. Deep learning models such as CNN, multilayer perceptron, or auto-encoders will be used. Our CIS will be described using the VHDL.

REFERENCES

[1] A. Singhal, M. Phogat, D. Kumar, A. Kumar, M. Dahiya, and V. K. Shrivastava, "Study of deep learning techniques for medical image analysis: A review," *Materials Today: Proceedings*, vol. 56, pp. 209–214, 2022, doi: 10.1016/j.matpr.2022.01.071.

[2] A. Shrestha and A. Mahmood, "Review of deep learning algorithms and architectures," *IEEE Access*, vol. 7, pp. 53040–53065, 2019, doi: 10.1109/ACCESS.2019.2912200.

[3] I. Negabi, S. E. Adib, S. A. Asri, and N. Raissouni, "Towards an intelligent cryptosystem design: deep learning cryptography for embedded systems security and connected objects," in *Colloque sur les Objets et systemes Connectes-COC'2021*, 2021.

[4] T. Kubota, K. Yoshida, M. Shiozaki, and T. Fujino, "Deep learning side-channel attack against hardware implementations of AES," *Microprocessors and Microsystems*, vol. 87, Nov. 2021, doi: 10.1016/j.micpro.2020.103383.

[5] J. Faraone *et al.*, "AddNet: deep neural networks using FPGA-optimized multipliers," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 1, pp. 115–128, Jan. 2020, doi: 10.1109/TVLSI.2019.2939429.

[6] H. A. Atee, R. Ahmad, N. M. Noor, and A. K. Ilijan, "Machine learning based key generating for cryptography," *Journal of Engineering and Applied Sciences*, vol. 11, no. 8, pp. 1829–1834, 2016, doi: 10.3923/jeasci.2016.1829.1834.

[7] C. Carlet, M. A. Hasan, and V. Saraswat, "Security, privacy, and applied cryptography engineering: 6th International Conference, SPACE 2016 Hyderabad, India, December 14–18, 2016 proceedings," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016, pp. 3–26.

[8] M. Ibrahim, A. Shaawat, and M. Torki, "Covariance pooling layer for text classification," *Procedia Computer Science*, vol. 189, pp. 61–66, 2021, doi: 10.1016/j.procs.2021.05.070.

[9] R. Gupta, S. Tanwar, S. Tyagi, and N. Kumar, "Machine learning models for secure data analytics: A taxonomy and threat model," *Computer Communications*, vol. 153, pp. 406–440, Mar. 2020, doi: 10.1016/j.comcom.2020.02.008.

[10] S. K. Rao, D. Mahto, and D. A. Khan, "A survey on advanced encryption standard," *International Journal of Science and Research (IJSR)*, vol. 6, no. 1, pp. 711–724, Jan. 2017, doi: 10.21275/ART20164149.

[11] I. Negabi and S. E. Adib, "Design of a reconfigurable cryptographic system on FPGA," (in French) in *Conception d'un système cryptographique reconfigurable sur FPGA, Colloque sur les Objets et systèmes Connectés, Ecole Supérieure de Technologie de Casablanca (Maroc), Institut Universitaire de Technologie d'Aix-Marseille (France)*, 2019.

[12] E. A. Samir and R. Naoufal, "Compact RIO based real time implementation of AES algorithm for embedded applications," *International Journal of Embedded and Real-Time Communication Systems*, vol. 10, no. 2, pp. 19–36, Apr. 2019, doi: 10.4018/IJERTCS.2019040102.

[13] A. Vuppala, R. S. Roshan, S. Nawaz, and J. Ravindra, "An efficient optimization and secured triple data encryption standard using enhanced key scheduling algorithm," *Procedia Computer Science*, vol. 171, no. 2019, pp. 1054–1063, 2020, doi: 10.1016/j.procs.2020.04.113.

[14] P. Kumar and S. B. Rana, "Development of modified AES algorithm for data security," *Optik*, vol. 127, no. 4, pp. 2341–2345, Feb. 2016, doi: 10.1016/j.ijleo.2015.11.188.

[15] A. A. Yazdeen, S. R. M. Zeebaree, M. M. Sadeeq, S. F. Kak, O. M. Ahmed, and R. R. Zebari, "FPGA implementations for data encryption and decryption via concurrent and parallel computation: a review," *Qubahan Academic Journal*, vol. 1, no. 2, pp. 8–16, Mar. 2021, doi: 10.48161/qaj.v1n2a38.

[16] S. E. Adib and N. Raissouni, "AES encryption algorithm hardware implementation: throughput and area comparison of 128, 192 and 256-bits Key," *International Journal of Reconfigurable and Embedded Systems (IJRES)*, vol. 1, no. 2, pp. 67–74, Jul. 2012, doi: 10.11591/ijres.v1.i2.pp67-74.

[17] N. Ahmad and S. M. R. Hasan, "A new ASIC implementation of an advanced encryption standard (AES) crypto-hardware accelerator," *Microelectronics Journal*, vol. 117, Nov. 2021, doi: 10.1016/j.mejo.2021.105255.

[18] P. Patel and A. Thakkar, "The upsurge of deep learning for computer vision applications," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 1, pp. 538–548, Feb. 2020, doi: 10.11591/ijece.v10i1.pp538-548.

[19] R. D. Bajaj, M. Gokhale, and M. T. Vlsi, "Design and simulation of AES algorithm for cryptography," *International Journal of Engineering Science and Computing*, vol. 6, no. 6, pp. 6340–6344, 2016.

[20] T. M. Kumar and P. Karthigaikumar, "FPGA implementation of an optimized key expansion module of AES algorithm for secure transmission of personal ECG signals," *Design Automation for Embedded Systems*, vol. 22, pp. 13–24, Jun. 2018, doi: 10.1007/s10617-017-9189-5.

[21] U. Arom-oon, "An AES cryptosystem for small scale network," in *2017 Third Asian Conference on Defence Technology (ACDT)*, Jan. 2017, pp. 49–53, doi: 10.1109/ACDT.2017.7886156.

[22] R. M. Jogdand and S. S. Bisalapur, "Design of an efficient neural key generation," *International Journal of Artificial Intelligence and Applications*, vol. 2, no. 1, pp. 60–69, Jan. 2011, doi: 10.5121/ijaia.2011.2105.

[23] S. C. Satapathy, B. N. Biswal, S. K. Udgata, and J. K. Mandal, *Proceedings of the 3rd International Conference on Frontiers of Intelligent Computing: theory and applications (FICTA) 2014*, vol. 328. Cham: Springer International Publishing, 2015.

[24] N. S. S. Srinivas and M. Akramuddin, "FPGA based hardware implementation of AES Rijndael algorithm for encryption and decryption," in *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, Mar. 2016, pp. 1769–1776, doi: 10.1109/ICEEOT.2016.7754990.

[25] S. P. Guruprasad and B. S. Chandrasekar, "An evaluation framework for security algorithms performance realization on FPGA," in *2018 IEEE International Conference on Current Trends in Advanced Computing (ICCTAC)*, Feb. 2018, pp. 1–6, doi: 10.1109/ICCTAC.2018.8370396.

[26] Y. Bentoutou, E.-H. Bensikaddour, N. Taleb, and N. Bounoua, "An improved image encryption algorithm for satellite applications," *Advances in Space Research*, vol. 66, no. 1, pp. 176–192, Jul. 2020, doi: 10.1016/j.asr.2019.09.027.

BIOGRAPHIES OF AUTHORS

Ismail Negabi received a bachelor's degree in electronics from the University of Sidi Mohamed Ben Abdellah (USMBA), Fes, Morocco. Holds a research master's degree in signal processing and machine learning from the National School of Applied Sciences, University of Abdelmalek Essaadi (UAE), Tetouan, Morocco, in 2017 and 2019, respectively. Currently, he is a Ph.D. student in mathematics-physics and new technologies at the Remote Sensing and Geographic Information System (RS and GIS) Laboratory, National School of Applied Sciences, University of Abdelmalek Essaadi, Tetouan, Morocco. His research interests include the design of intelligent cryptosystems based on deep learning modules. He can be contacted at ismail.negabi@etu.uae.ac.ma.

Smail Ait El Asri received a bachelor's degree in electronics and industrial computer science from the University of Moulay Ismail (UMI), Meknes, Morocco. Holds a research master's degree in signal processing and machine learning from the National School of Applied Sciences, University of Abdelmalek Essaadi (UAE), Tetouan, Morocco, in 2017 and 2019 respectively. Currently, he is a Ph.D. student in mathematics-physics and new technologies at the Remote Sensing and Geographic Information System (RS and SIG) Laboratory, National School of Applied Sciences, University of Abdelmalek Essaadi, Tetouan, Morocco. His research interests include the design of an intelligent system for the automatic detection of buildings on very high-resolution satellite remote-sensing images. He can be contacted at smail.aitelasri@etu.uae.ac.ma.

Samir El Adib received a degree in Informatics, Electronics, Electrotechnics, and Automatics (IEEA) and an M.S. degree in automatic and data processing from University Abdelmalek Essaadi (UAE), Tetuan, Morocco, in 2004 and 2006 respectively. respectively. He has been a professor of physics and remote sensing at the National Engineering School for Applied Sciences of the UAE of Tetuan since 2015. Currently, he is a member of the Remote Sensing and GIS Laboratory. His main research interests are FPGAs in custom-computing applications, and more concretely, applications of reconfigurable hardware to cryptography. He can be contacted at adibsamir@gmail.com.

Naoufal Raissouni received an M.S. and a Ph.D. degree in physics from the University of Valencia, Spain, in 1997, and 1999, respectively. He has been a professor of physics and remote sensing at the National Engineering School for Applied Sciences of the University Abdelmalek Essaadi (UAE) of Tetuan, since 2003. He is also heading the Remote Sensing and GIS Lab in the UAE. His research interests include atmospheric correction in visible and infrared domains, the retrieval of emissivity and surface temperature from satellite images, huge remote sensing computations, mobile GIS, ad hoc networks, and the development of remote sensing methods for land cover dynamic monitoring. He can be contacted at naoufal.raissouni.ensa@gmail.com.