Implementation of a Hardware Accelerator for a Real-time Encryption System

Islam Mohamed Shaher[1], Moustafa Mahmoud[2], Hassan Ibrahim [1], Moustafa Ali [3], Hassan Mostafa [1,4]

[1]Electronics and Communications Engineering Department, Cairo University, Giza 12613, Egypt.
[2]Electronics and Communications Engineering Department, Alexandria University, Alexandria 11432, Egypt.
[3]Electronics and Communications Engineering Department, Ain Shams University,Cairo, 11566, Egypt.
[4]Nanotechnology Department at Zewail City for Science and Technology, Cairo, Egypt.
{*islam.shaher@gmail.com, e.moustafa2@gmail.com, hassan.ibrahim98@hotmail.com*
eng.moustafa.ali.2015@gmail.com, hmostafa@uwaterloo.ca}

***Abstract*—IoT devices are spreading everywhere and its progress increases day by day. The security of these devices must be the biggest concern. This paper is describing a complete system of a surveillance camera transceiver using two PYNQ Boards connected through Ethernet cable. On one side there is a camera, on the other side there is an HDMI screen and in between is the proposed software image processing and encryption/decryption using the Ascon algorithm (one of the winners of CAESAR Competition for the lightweight category). The aim is to get low power, low area, and low-cost design implementation.**

***Index Terms*—IoT (internet of things), CAESAR Competition, PYNQ Board, ZYNQ (Zynq-7000 SoC).**

I. Introduction

IoT Applications spread rapidly, and it is expected that there will be more than 50 billion 'things' connected to IoT [1]. on the other side, The manufactures of IoT devices have not taken the security implementation seriously in the devices. In 2013, Doctors disabled wireless in Former US Vice President Dick Cheney's pacemaker to thwart hacking [2]. Also, there are a lot of examples of unsecured IoT devices that can make disastrous problems [3]. Besides, the software technique to secure IoT devices is not sufficient:

- It requires too much disk space which is not practical for IoT applications [4].
- The operating systems trusted the hardware devices so the need for identifying the device IDs is so important.
- It can't be updated in most cases.

The traditional Hardware techniques consume large power and its area will decrease the device resources. Hardware security requires a small area and low power consumption to be applied for IoT devices. CAESAR competition is targeting lightweight encryption algorithms as a first aim. A lot of implementations have been done addressing energy-limited applications [8] [9] . The winners in the final portfolio are Ascon and ACORN for lightweight applications [5]. In this paper, an accelerator for the ascon encryption algorithm [6] [7] for software interfaces is implemented . The algorithm is chosen based on analysis done in [15] and [10]. VHDL language is used in the implementation using an improved high-speed implementation with a lot of modifications on the control and datapath modules to suit axi4 protocol. This study is to implement a complete system using a modified Ascon algorithm on PYNQ Board [11], this board can be programmed using python. Python is used for interactions with the PS (Processing Subsystem) part of Zynq. In our system, the transmitter is used to compress and encrypt the image for fast and secure communication with the receiver. At the receiving end, the reverse process occurs, where decryption then image decompression occurs. The last step is to display the image on a monitor. Two PYNQ Boards are used for a full demonstration of the complete hardware secured system.

II. Methods

The image is captured using a webcam, there are no constraints on the used webcam, however, the lower the resolution the faster the system will get. The previous point introduces a trade-off between image quality, speed, and power, as the resolution increases, speed degrades and the power increases. In this implementation, the used resolution is (640, 480).
The second stage is image compression. In this stage, the same trade-off appears.
The next stage is to encrypt the compressed image. The encryption algorithm used is Ascon which is symmetric (the encryption and decryption are done using the same hardware). Therefore, this stage is identical to the decryption stage. There are various implementations of the Ascon algorithm [12]. The two implementations tested are:

- Iterated implementation (without any optimizations).
- Loop unrolling implementation.

The preceding step is to send the compressed and encrypted image to the receiving side. There are numerous protocols and techniques to send data. In our case, the Ethernet protocol is used. The advantages of Ethernet are:

- Simple implementation

- Relatively fast speed

However, the main drawback of Ethernet is that it is a wired connection.
Finally, when the data is received, it is decrypted (using the same hardware of encryption) then decompressed and displayed on the monitor using HDMI.

A. IMAGE PROCESSING using Python OpenCV

Encrypting images with their original sizes captured from a webcam takes a long time. Since the system is required to be a real-time one, The time taken is required to be reduced. The method used to do so is to compress the captured images before encryption and sending them to the receiver.

B. Image compression before encryption and sending

OpenCV imencode function is used to compress the image and store it in the memory buffer, this compression process is done according to image type then the imencode function will choose the codec according to the extension [13].
JPEG typically achieves 10:1 compression with perceptible loss in image quality. On the other hand, the PNG format sustains its quality when it is compressed and decompressed. The compression function reduces the image size by nearly 15% of its original one.

C. TRANSMITTING ENCRYPTED IMAGE BETWEEN TWO FPGA Boards

The encrypted data is transmitted using TCP/IP protocol (Transmission Control Protocol/Internet Protocol). PYNQ framework facilitates using TCP protocol using python that already has the built-in function (Socket) to build network programs and scripts to transfer data between transmitter and receiver using TCP protocol [14].

D. ASCON ALGORITHM IMPLEMENTATION

The following figure Fig. 1 shows the hierarchy of our FPGA implementation of hardware acceleration using the Ascon algorithm. This acceleration is applied to a real-time video surveillance application between two SOC PYNQ FPGA Boards.

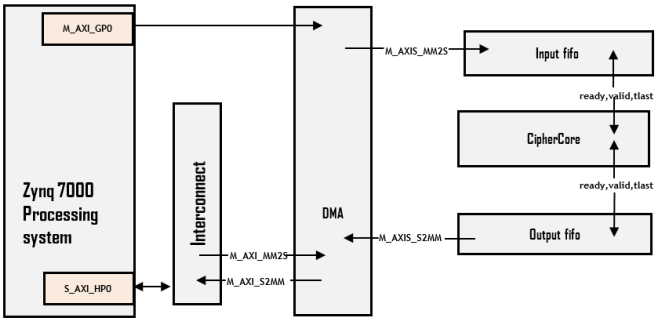

Fig. 1. **FPGA Implementation Hierarchy**

The implementation was divided into separate blocks using the Vivado design tool it consists of:

- The cipher core IP using programmable logic.
- Dual arm a9 processors.
- DMA between DRAM and PL (Programmable logic).
- HDMI IP to show final results.

Zynq processing unit has GP0 which is a general-purpose AXI master port so that it can configure the DMA using AXI lite to set up the transfers and trigger them. The other interface is the HP0 port is a high-performance AXI slave port that is needed to access the DDR controller to allow the DMA to read and write from the DDR. DMA is used to pass the data directly between the DRAM and the programmable logic through the input and output FIFO using AXI4 Stream where the cipher core makes synchronization with the two FIFO modules using the three signals (valid, ready, tlast) where tlast denotes the end of plaintext and is used in the output side to free the bus and make it ready for a new reception. AXI Lite was used to sending initialization to the core such as the key, pub, and associated data. AXIS label in the diagram represents an AXI stream bus while the AXI label denotes an AXI lite bus. MM2S represents the data directly from a memory mapped device to the streaming side which is the cipher core.

1) **Summary of the encryption algorithm:** The following describes the encryption flow from the beginning to the production of ciphertext and authentication tag:

- The state is initialized using key and npub and passes by 12 round transformation, then the final result is xor-ed with the key.
- The state is XORed with the 64 bits of npub data and passes by 6 round transformations this process is repeated “m” / 8 times where m is the length of npub in bytes.
- The state is XORed with the 64 bits of plaintext and passes by 6 round transformations this process is repeated “m” / 8 times where m is the length of plaintext in bytes. The last chunk of data is not transformed after being XORed.
- In the finalization state, the state is XORed with the key and passes by 12 round transformations. The least bit of the final result is XORed with the key and is sent as an authentication tag.

In each round transformation, the input data (if exist in the stage) is xor-ed with the first 64-bit of the state and the output is fed into the round transformation with the remaining 256 bits of the state without being changed. The first layer is doing xor to the input with around constant which changes according to the round order.
The second substitution layer is called the S-box stage where the state is divided into 5 64-bit chunks. Each nth bit of every chunk is entered into the S-Box transformation where each possible input of the 32 is mapped to a different output. The third layer is called the linear diffusion layer where the 320-bit output of the S-box layer is being xor-ed with two rotated versions of itself. This process is non-invertible. Rotation is more secure than shifting operation as Xor operation with two shifted versions is invertible hence any known intermediate state can be used to get the older state which may leak secret

information.

2) **hardware implementation:** The Ascon algorithm has a higher LUTs utilization than ACORN, The comparison led us to choose Ascon since it has a better data processing rate that suits our application with a close power consumption compared to ACORN [15] .
The proposed implementation design the control and data-path modules of the cipher core so that it can be easily integrated with AXI4 Streaming FIFO for high data rate transmission. The encryption process does not wait for the end of filling the FIFO but is processing the data as long as the FIFO is not empty, so parallel processing is done. the investigating started by the possible implementations can be made to get the maximum performance. There are three main stages in every round transformation.

These stages can't be pipelined since the first stage which is adding around constant can't process new data at the same time while the second stage S-box is working which is a necessary condition for pipe-lining. This is because it must wait for the output of the third stage.

Hence, another type of optimization shows up as unrolling the loop can be helpful. The investigation was in two modes of unrolling using 3 and 6 unrolling factors. It is found that the length of the critical path does not increase by a factor of 3 (or 6) since the synthesis tool makes further simplification and optimization producing a better input-output relation and using higher-order n-input LUTs.

The ratio between the new and old maximum frequency is higher than 1/3 (or 1/6) so the throughput increase for higher unrolling factor where all are compared at the maximum frequency for each of them. The gain in the throughput can be determined by the following equation:

$$Throughput\_Gain = \frac{NewFreq}{Old\,Freq} * UnrollingFactor \quad (1)$$

Which is always greater than 1.
The increase in throughput by increasing the unrolling factor is clear in the next figures Fig. 2 and Fig. 3. where the increase in the gain decreases for higher unrolling factors which can be noticed in the transition of the factor from 3 to 6. The comparison between the 4 implementations in terms of power, maximum frequency, utilization, and throughput are made. The throughput mentioned in the figure is the throughput of the encryption core using the programmable logic before the integration with the ZYNQ processing block.
There is a lot of possible requirements that determine the best implementations to choose according to the type of application, this study choosing the high throughput implementation with an unrolling factor of 3 since the application requires real-time processing of video frames. the implementation with a factor of 6 increases the utilization without a noticeable rise in the throughput.

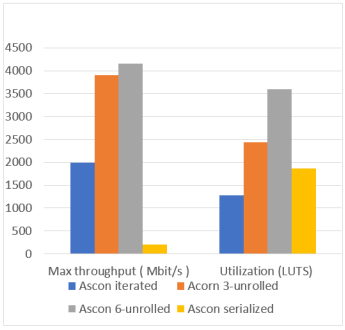

Fig. 2. **Maximum Throughput and Utilization for iterated, 3-Unrolled, 6-unrolled, serialized**

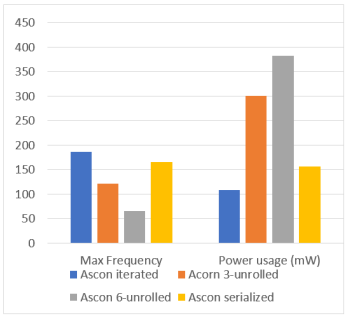

Fig. 3. **Maximum Frequency and Power Usage for iterated, 3-Unrolled, 6-unrolled, serialized**

III. Results

A. Image Encryption Results

Fig. 4 is the encrypted image where the complete image is encrypted as one block of data without reentering the initialization state. Figure Fig. 5 shows the encrypted image by repeating the encryption from the beginning of the initialization state for every 8 bytes. This means that if two consecutive 8 pixels are exactly the same, the ciphered pixels will be the same. This leaks some information about the original image which is clear in the figure. Security requirements of the algorithm stated that encryption of the same data should be accompanied by changing the public number which will obviously resolve the problem due to changing the state xored with the plain-text as a result of changing some content of the initialization state.

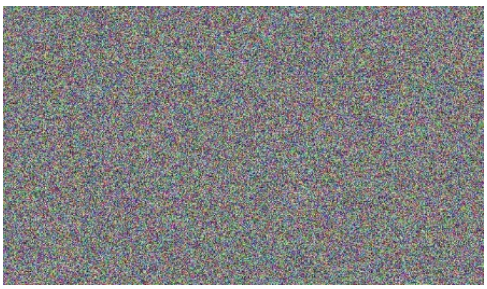

Fig. 4. **Encrypted image**

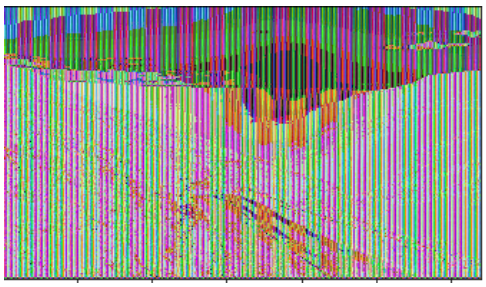

Fig. 5. **Encrypted image without changing public number and repeating every 8 bytes**

1) **Performance parameters of the system :** This Section includes the performance parameters in Table. I.

TABLE I
PERFORMANCE PARAMETERS

Throughput of the encryption core	488 MBYTES/Second
Throughput of AXI4 linkage between the processor and PL	133 Mbytes/Second
Length of the original image	695 KBytes
Size of the image after jpeg compression	24 KBytes
Throughput of the encryption system including latencies resulted from image processing ,compression and linkage between the two boards	455.779 KBYTES/Second
Overall throughput of a video surveillance system	299.118 KBYTES/Second
FPS	13 FPS

2) **Comparison with a fully software implementation:** In this section, it is assumed that the Ciphertext is required to exist in a Linux environment for many purposes like file encryption or being sent over a network between a server and a client. The full software implementation is to be compared with the SoC one. Hence, the SoC implementation throughput to be mentioned is the one including the overhead produced due to the bus speed connecting the encryption core IP and the ZYNQ processing unit. Both implementations were tested on the same ARM Cortex A9 processor. The results are shown in Table. II.

TABLE II
COMPARISON WITH SOFTWARE IMPLEMENTATION

Software implementation throughput	33.646 KBYTES/Second
SoC implementation throughput	455.779 KBYTES/Second

IV. CONCLUSION

In this study, a proposed complete IoT system for surveillance applications. The application includes image processing to reduce the image size using the OpenCV library. The encryption module is a modified version of the Ascon algorithm to fit in this application. PYNQ board is a cheap resource helps us to use all python facilities. The comparisons between the software implementation and the one with a hardware accelerator show that the hardware-accelerated design can perform many times better than the pure software one. Future work will be improving the speed of the system by using an optimized software interface to avoid affecting the high throughput gained from the hardware accelerator.

V. ACKNOWLEDGMENT

This work was supported by the Egyptian Information Technology Industry Development Agency (ITIDA) under ITAC Program PRP2018.R25.23.

REFERENCES

[1] CompTIA, “Sizing Up the Internet of Things”, https://www.comptia.org /resources/sizing-up-the-internet-of-things, 2015.
[2] BBC News, ”Dick Cheney: Heart implant attack was credible, ” https://www.bbc.com/news/technology-24608435, 2013.
[3] CNN, ”FDA confirms that St. Jude’s cardiac devices can be hacked, ” https://money.cnn.com/2017/01/09/technology/fda-st-jude-cardiac-hack/
[4] H. Suo, J. Wan, C. Zou and J. Liu, ”Security in the Internet of Things: A Review,” 2012 International Conference on Computer Science and Electronics Engineering, Hangzhou, 2012, pp. 648-651.
[5] Bernstein, ”Cryptographic competitions, ” https://competitions.cr.yp.to/ caesar-submissions.html, 2019.
[6] Dobraunig, Christoph, et al. ”Ascon v1. 2. submission to the CAESAR competition.” CAESAR First Round Submission, 2014.
[7] Gross, Hannes, et al. ”Ascon hardware implementations and side-channel evaluation.” Microprocessors and Microsystems, 2017.
[8] A. Abbas, H. Mostafa, and A. N. Mohieldin, “Low Area and Low Power Implementation for CAESAR Authenticated Ciphers”, Journal of Low Power Electronics (JLOPE), vol. 15, no. 1, pp. 104-114, 2019.
[9] S. Soliman, M. A. Jaela, A. M. Abotaleb, Y. Hassan, M. A. Abdelghany, A. T. Abdel-Hamid, K. N. Salama, and H. Mostafa, ”FPGA Implementation of Dynamically Reconfigurable IoT Security Module Using Algorithm Hopping”, Elsevier Integration VLSI Journal, vol. 68, pp. 108-121, 2019.
[10] S. Sharaf, and H. Mostafa, “A Study of Authentication Encryption Algorithms(POET, Deoxys, AEZ, MORUS, ACORN, AEGIS, AES-GCM) For Automotive Security”, IEEE International Conference on Microelectronics (ICM 2018), Sousse, Tunisia, pp. 315-318, 2018.
[11] PYNQ , ”What is PYNQ?, ” http://www.pynq.io/.
[12] Groß, Hannes, et al. ”Made-to-Measure Hardware Implementations of ASCON.” 2015 Euromicro Conference on Digital System Design. IEEE, 2015.
[13] OpenCV,”Open Source Computer Vision, ” https://docs.opencv.org/3.4/.
[14] Python, ”Low-level networking interface, ” https://docs.python.org/3 /library/socket.html.
[15] N. Samir, A. S. Hussein, M. Khaled, A. N. ElZeiny, M. Osama, H. Yassin, A. Abdelbaky, O. Mahmoud, A. Shawky, and H. Mostafa, “ASIC and FPGA Comparative Study for IoT Lightweight Hardware Security Algorithms”, Journal of Circuits, Systems, and Computers (JCSC), vol. 28, no. 12, pp. 1-13, 2019.