VFAST Transactions on Software Engineering
http://vfast.org/journals/index.php/VTSE@ 2020 ISSN(e): 2309-3978;ISSN(p): 2411-6246
Volume 8, Number 1, January-December, 2020

pp. 08-13

DESIGN AND IMPLEMENTAION OF SECURE VIDEO COMMUNICATION OVER IP USING PYNQ-Z1 BOARD

Aniq ur Rahman[1], Luqman Abbas[2], Javaid khurshid[3], Roveed Ahmed[4], Muhammad Hanif Durad[5]
[1,3,5]Department of Computer and information Sciences, Pakistan Institute of Engineering and Applied Sciences, Islamabad, Pakistan
[2s,4]CESAT, The National Center for Physics, Islamabad, Pakistan

Abstract— Now a day's security cameras are everywhere and are important part of surveillance system. Specifically, in sensitive areas cameras are essentials. So now security is a big concern for surveillance system itself. Through covert eavesdropping Communication of surveillance systems can be tapped with various techniques this is often called bugging. A "Bug" is a device that is placed in an area, which then intercepts communications and transmits or conducts them out of that area to a listening post. Most of the organization didn't pay much attention on this vulnerability. So, in this paper solution is presented for securing communication by using pynq-z1 FPGA board.
Keywords— FPGA, APSoC, processing system, programmable logic, Python, Zynq, PYNQ, TMDS, IP, AXI, HDMI.

1. Introduction. Video surveillance systems are important part of security systems in sensitive areas. There are many threats to video transmission e.g. Side channels attacks, covert channels attacks. In side channel attacks attacker finds the vulnerability in computer system rather than in implemented algorithm. e.g. timing information, electromagnetic leaks, sound or even power consumption can provide extra source of information which intruders can exploit. To prevent these attacks video transmission should be secured so the purpose of this work is to have a secure communication over internet protocol (IP) using PYNQ-Z1 FPGA board. So, for this project pynq board is used. The PYNQ-Z1 board is designed to be used with PYNQ, a new open-source framework that enables embedded programmers to exploit the capabilities of Xilinx Zynq All Programmable SoCs (APSoCs) without having to design programmable logic circuits. Instead the APSoC is programmed using Python, with the code developed and tested directly on the PYNQ-Z1. The programmable logic circuits are imported as hardware libraries and programmed through their APIs in essentially the same way that the software libraries are imported and programmed which makes it very easy to program[1]. PYNQ-Z1 is the first APSoC which is programmed using python. Video overlay for the board designed in Xilinx vivado and can be used for any other zynq board for improvement. For video input and output to the board HDMI interfaces are used. The input video is then processed and encrypted and sent to other board via ethernet interface. It is also ensured in this project that encryption should be lossless and real time. So that it could easily be decrypted at receiver end for display in real time.

2. Related work. The best way for the protection of communication of video surveillance system is to use end to end encryption. Encryption assures the confidentiality of data being transferred. A lot of research is ongoing to find the best algorithm for encrypting video stream. In the following sections we discuss two cryptographic algorithms approaches that are widely used. And we will compare their efficiency and effectiveness in video stream data.

A. **Naïve algorithms:** Conventional cryptographic algorithms include, public key algorithms, shared secret key algorithms and irreversible hash functions each has its advantages and disadvantages. In shared secret key there are two types block ciphers and stream ciphers. When AES succeeded DES, it was considered secure cipher and efficient in hardware and software implementation[2]. But after the emerging of stream ciphers they left behind the block ciphers for real time processing applications.In public key cryptography algorithms cannot be used interchangeability. DSA (Digital signature algorithms) used for digital signature creation, Diffie-Hellman used for key exchange, RSA can be used for key exchange, digital signature creation and encryption. While hash function algorithms are used for integrity of message to ensure that message is not altered during transmission.

B. ***Video encryption algorithms****:*With the advancement in digital image capturing technology frame rate of higher quality has increased than that has been seen before and this trend continues. So real-time encryption of high-quality frames coming with faster data rate is hard to achieve. So, video specific encryption techniques are proposed to achieve real time encryption. Researchers has proposed different techniques to increase the speed of naïve algorithms. Refer to Singh and Manimegalai in [3] for detailed compression and encryption techniques for video data. Video encryption algorithms are generally classified based on their compression techniques.

One is joint compression and encryption algorithms in that algorithm's encryption is applied during one of the compression stages. while other are compression independent algorithms in which encryption is applied before compression or after compression so that are compression independent. In first scheme of algorithms they have the drawback related to video encoder and decoder. During the implementation they change the CODEC, so the video cannot be playback with the original CODEC and has to used special equipment to playback the video and they due to susceptible to chosen and known plaintext attacks their security efficiency is reduced and cannot be used for encryption. While the second scheme algorithms are faster are reduced the overhead by 90% comparative to naïve algorithms, but it is vulnerable to known plaintext attack[4].

So above mentioned video encryption methods have strengths and weaknesses against naïve algorithm. They have less overhead compared to naïve algorithm. But they alter the codec during encryption and these algorithms are vulnerable to attacks and offer weaker security. Therefore, none of these video encryption algorithms are included in information system security standards. So, these algorithms cannot be used for security sensitive applications. Therefore, we chose the salsa20 stream cipher for encrypting video stream which is much secure than any block cipher e.g. AES. A complete SAT solver returns unsatisfiable which means that no differential characteristics exists for salsa20 stream cipher, so it is secure against differential and linear cryptanalysis attacks. And it is also computationally efficient on both hardware and software platforms.

3. prototype implementation. The main theme is to secure communication of video surveillance system over internet protocol (IP) using PYNQ board. The block diagram for this is shown in figure 1. For video input and output HDMI interface on PYNQ board is used.

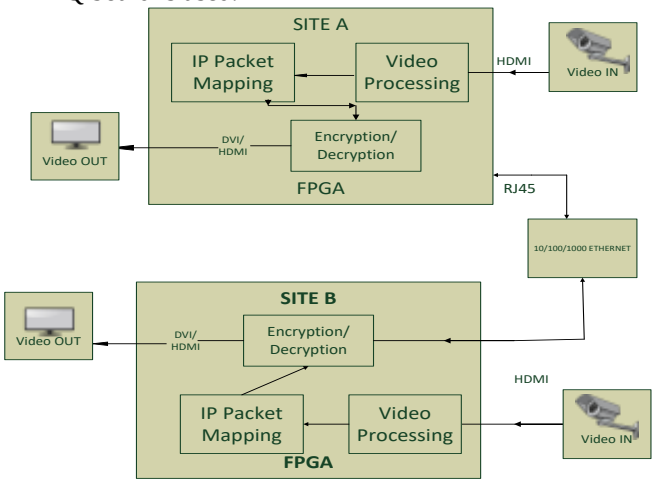

Figure 1 implemented design for secure video communication

The PYNQ-Z1 comprises two unbuffered HDMI ports: one sink port (input), and one source port (output). HDMI uses Transition-minimized differential signaling (*TMDS*) protocol which is implemented in video overlay in Vivado. The IP for HDMI input shown in figure 2 and IP for HDMI out shown in figure 3. Then these interfaces can be accessed within PS with Python drivers. Video frames received from HDMI input and then processed in PS. The HDMI-in IP as shown in figure 15 can capture predefined standard HDMI

resolutions. After connecting HDMI source and the HDMI input controller for the HDMI-in IP is started, Incoming data will be automatically detected. Functions for HDMI written in python, so resolution can also be read from the HDMI Python class. In frontend IP different IP's can be seen. Dvi2rgb IP interfaces directly to raw TMDS data channel inputs and clock as defined in DVI 1.0 specifications for input devices. By recovering TMDS[5] stream dvi2rgb IP translates the video stream and outputs the 24-bit RGB video data along with the synchronization signals and pixel clock[6].

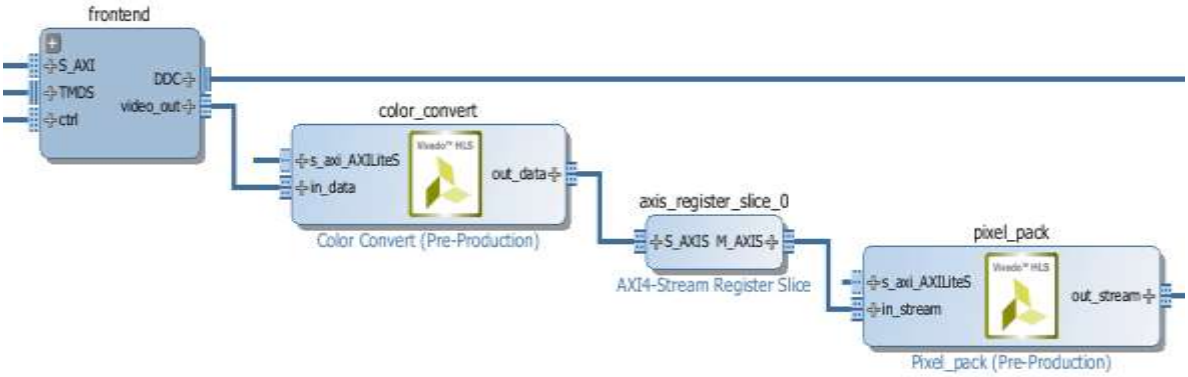

Figure 2 HDMI input IP

Color convert IP simply converts RGB data format to RBG format[7]. Then video in to AXI 4 stream IP converts parallel video signals[8] e.g. DVI video signals to AXI4 stream data and the image data can be streamed to the PS DRAM.

With this video designers can easily and quickly connect an external video source to next processing blocks on the AXI4-Stream interface that uses a video protocol. This core works in combination with the Xilinx Video Timing Controller (VTC) IP core[9] to detect features of the incoming video format which is read by system processor and then used to configure next processing blocks.

Video out from the frontend then sent to the color convert IP. The color convert and pixel pack IP allow color conversion between different spaces at runtime.

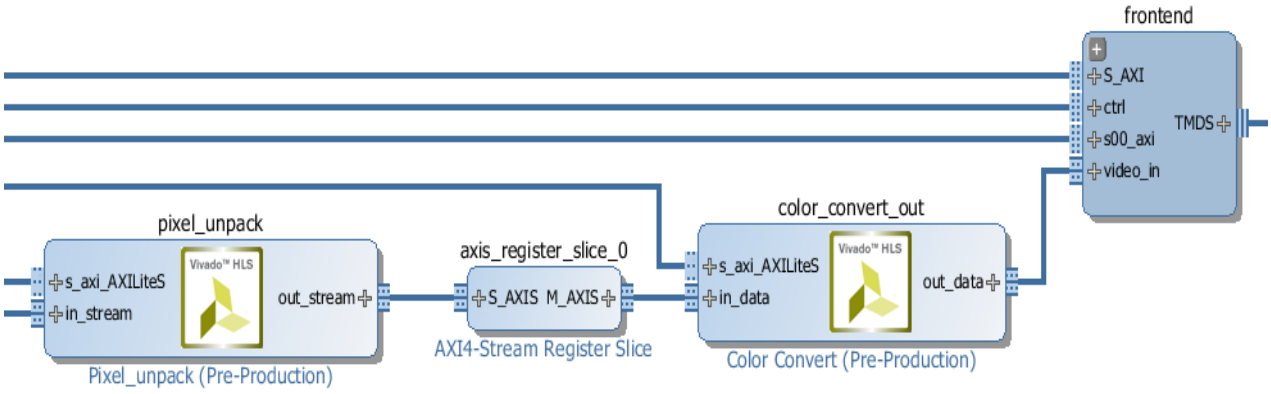

Figure 3 HDMI out IP

HDMI out IP is similar to HDMI in IP just difference is that HDMI in uses pixel pack IP while HDMI out uses pixel unpack IP. And the IP's in hierarchy are connected in reverse order[10].

Now when video frames are available for further processing encryption is applied for securing communication. For real time encryption of frames encryption algorithm should be fast and secure. And stream ciphers are most suited for real time processing. So frames are encrypted with salsa20/20r stream cipher[11]. Salsa20 is a stream cipher and computationally its performance is better than other ciphers, and it is also cryptographically strong. And it is also compatible on hardware and software platforms. Its fast and small and so beneficial in less computational resources. The current best differential attack on Salsa20 is on eight rounds though key taken into consideration is 128 bit key instead of 256 bit key[12]. Salsa20 cipher is also implemented in PS. The basic function of salsa20 cipher is shown in figure 4.

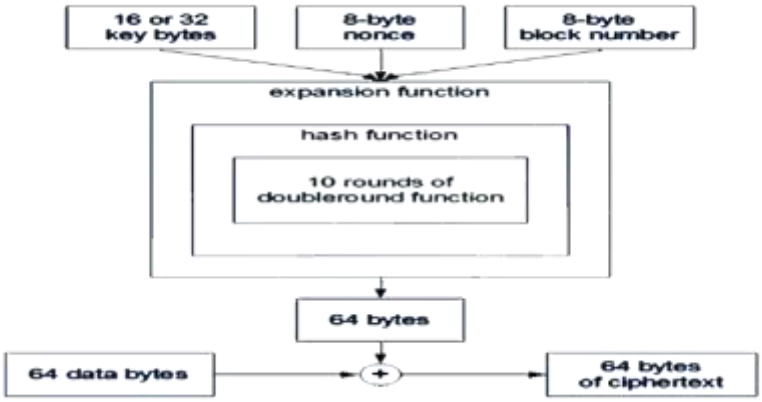

Figure 4 Salsa20 operation

One of the most important properties of cryptographic systems is its proof of security. So firstly, Entropy values for salsa20 cipher images are given in tables 1.

TABLE I. ENTROPY VALUES FOR SALSA20[13]

cipher	*Rounds*	*Entropy value*
Salsa20 cipher	8	7.9969
	12	7.9970
	20	7.9971

Secondly, to prevent the leakage of information it is very important to ensure that histogram doesn't have similarities which can be seen in figures 5(b), 5(d), 5(f), 5(h), 5(j) and 5(l) which are the histograms of figures 5(a), 5(c), 5(e), 5(g), 5(i) and 5(k) respectively.

Figure 5(a) Plane image

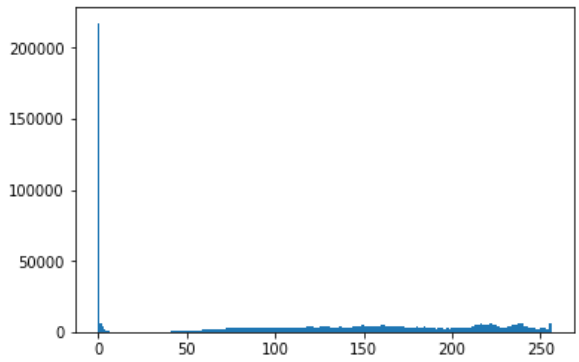

Figure 5(b) Histogram of plane image

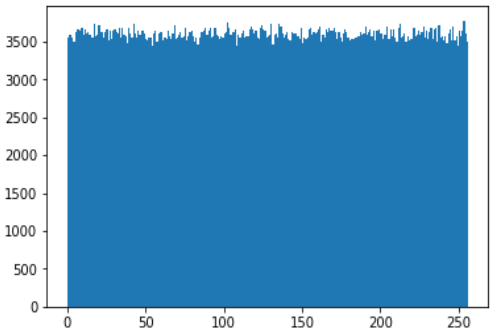

Figure 5(c) Encrypted image

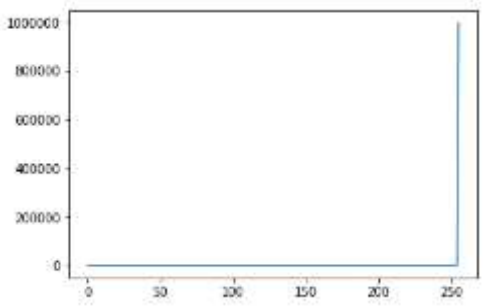

Figure 5(d) histogram of encrypted image

Figure 5(e) white frame

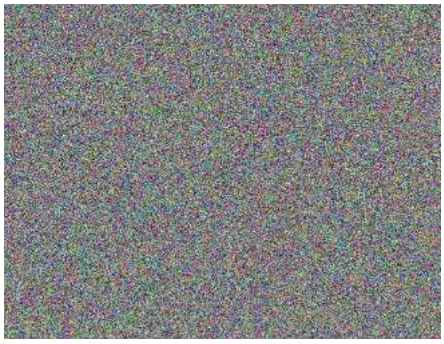

Figure 5(f) histogram of white frame

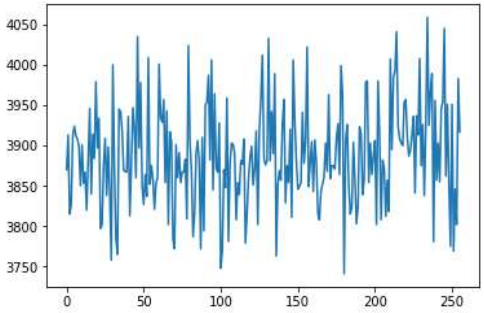

Figure 5(g) Encrypted white frame

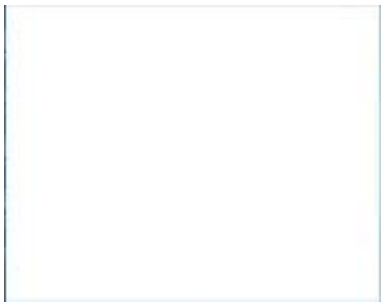

Figure 5(h) hist of fig 5(g)

Figure 5(i) black image

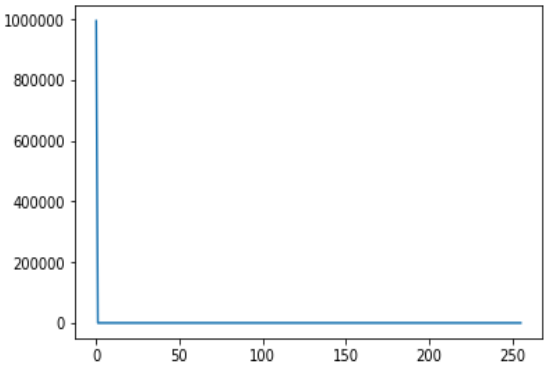

Figure 5(j) hist of black image

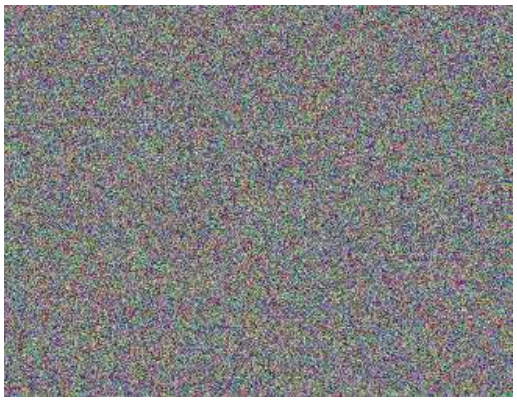

Figure 5(k) Encrypted fig 5(i)

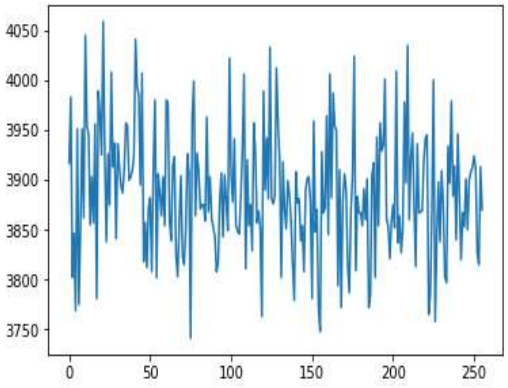

Figure 5(l) hist of fig 5(k)

4. Conclusion and future research. We've developed the prototype for protection of communication of video surveillance system over IP and presented our work in this paper. we've encrypted real time video on APSoC ZYNQ-7020 FPGA on PYNQ board. For that purpose, video overlay is designed in Xilinx vivado which can be used for any of ZYNQ FPGA. Video input interfacing has been achieved and frames are accessible in PS for video processing. Then frames are successfully encrypted and decrypted with best cipher suite. It takes 4ms to encrypt a frame of 480p in processing system.

Although all modules are implemented but there is room for improvement in the work. Firstly, the custom encryption IP for salsa can be designed and inserted in Video overlay to improve performance. Secondly, Network Input/output processor (IOP)[14] could be designed and by modifying Linux kernel driver the communication over ethernet could be speed up.

ACKNOWLEDGMENT

The authors would like to thank to PIEAS for funding this project, Dr Hanif durad from PIEAS and Mr. Salman Ahmed from CESAT for helping out in project.

REFERENCES

[1] Wang, Y., Feng, B., Li, G., Deng, L., Xie, Y., & Ding, Y. (2019). AccD: A Compiler-based Framework for Accelerating Distance-related Algorithms on CPU-FPGA Platforms. *arXiv preprint arXiv:1908.11781*.

[2] Perlman, R., Kaufman, C., & Speciner, M. (2016). *Network security: private communication in a public world*. Pearson Education India.

[3] Singh, K. J., & Manimegalai, R. (2012). A survey on joint compression and encryption techniques for video data. In *Journal of Computer Science*.

[4] Liu, F., & Koenig, H. (2010). A survey of video encryption algorithms. *computers & security*, *29*(1), 3-15.

[5] Qu, Y., Chen, Y., & Chen, W. (2017, November). Co-design and Implementation of Image Recognition Based on ARM and FPGA. In *National Conference on Embedded System Technology* (pp. 141-153). Springer, Singapore..

[6] *Contribute to Digilent/vivado-library development by creating an account on GitHub*. Digilent, 2018.

[7] "RGB to YCrCb Color-Space Converter." [Online]. Available: https://www.xilinx.com/products/intellectual-property/rgb_to_ycrcb.html. [Accessed: 15-Oct-2018].

[8] "AXI Reference Guide," vol. 13, p. 82, 2011.

[9] "Video Timing Controller v6.1 LogiCORE IP Product Guide (PG016)," p. 90, 2017.

[10] "Zynq-7000 SoC." [Online]. Available: https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html. [Accessed: 20-Sep-2018].

[11] Bernstein, D. J. (2008). The Salsa20 family of stream ciphers. In *New stream cipher designs* (pp. 84-97). Springer, Berlin, Heidelberg.

[12] Mouha, N., & Preneel, B. (2013). *Towards finding optimal differential characteristics for ARX: Application to Salsa20*. Cryptology ePrint Archive, Report 2013/328.

[13] Jolfaei, A., & Mirghadri, A. (2010). Survey: image encryption using Salsa20. *International Journal of Computer Science Issues (IJCSI)*, *7*(5), 213.

[14] *Networking Overlay on PYNQ. Contribute to Xilinx/PYNQ-Networking development by creating an account on GitHub*. Xilinx, 2018.