Hardware Acceleration of AES Cryptographic Algorithm for IPsec

BENHADDAD Omar Hocine, SAOUDI Mohamed, DROUICHE Amine, RABIAI Mohamed, ALLAILOU Boufelidia

Department of Electronics, ESACH, Algiers, Algeria
Corresponding author: omar.hocine.benhaddad@gmail.com
Received: 03/05/2019, Accepted: 13/08/2019

Abstract
This research considers the offloading of the IPsec packet encryption algorithm into an FPGA by proposing a hardware acceleration of the AES cryptographic algorithm for IPsec. We point out the benefits of relying on HW acceleration in terms of speed and energy efficiency for applications like IPsec. We present the description of the architecture of the proposed solution, the simulation results of our implementation of the AES algorithm in ECB (Electronic Code Book) mode. We also present the integration of the encryption core with the IPsec subsystem through a PCIe bus interface so that the resulting implementation is interoperable with other systems.

Keywords: IPsec, FPGA, AES, RIFFA, cryptography.

Introduction
The volume of data exchanged over networks has been increasing over the past decade. An increase that goes along with the need to protect this data at a much higher rate as it is more and more exposed on public networks. Sensitive data should be secured seamlessly and the process shouldn’t act as a bottleneck.

In 1998, IPsec was developed by the IETF (Internet Engineering Task Force). It is now one of the most popular solutions to secure data at the IP layer. The IPsec protocol is almost always integrated into the TCP/IP stack on Operating Systems (Windows, Linux...) but it needs a lot of computational power. One of the most computationally intensive parts is the encryption algorithm which in our case is the AES (Advanced Encryption Standard).

In this paper, we consider the feasibility of offloading AES cryptographic algorithm from software to FPGA connected through a PCIe endpoint to a host computer running on a Linux distribution.

Related Works
Several papers has been published in the field of hardware acceleration of cryptographic algorithms. Paillier & Verbauwhede (2007) achieved a 15.3 Gbps from their implementation of the AES in GCM mode on a Virtex-4 FPGA under a clock rate of 120 Mhz. In 2011, the implementation of the AES algorithm by Soliman et al. (2011) reached 74 Gbps on a Virtex-5
FPGA under the clock rate of 557 Mhz. In 2016, Smekal et al. (2016) described the AES implementation on Virtex-7 that achieved a 5.1 Gbps throughput under a clock rate of 100 Mhz. In 2013, Yun Niu et al (2013) implemented a design that includes 8 IPsec protocol IP cores and 24 crypto IP cores. The design give the throughput of 11.28 Gbps under a clock rate of 300 Mhz.

**IPSEC**

IPsec (Internet Protocol Security) is a set of protocols that uses cryptographic algorithms to ensure private and protected communications on IP networks. It operates on the network layer (3rd layer of the OSI model), therefore preventing the user from reconfiguring applications with IPsec standards on the application layer.

The main purpose of IPsec is to authenticate and to encrypt the data flow between the two participants in order to ensure confidentiality and integrity. In order to establish an IPsec connexion we need first to exchange the keys through the IKE protocol (Internet Key Exchange) which is used to authenticate the two participants of a secured tunnel by exchanging shared keys. Then the transfer of the data flow can be done through two possible protocols which are AH (Authentication Header) or ESP (Encapsulating Security Payload).

![Figure 1](image_url)

**Figure 1.** Communication between IPsec in the host computer part and the AES in the FPGA part through PCIe bus

The Figure 1 shows the position of IPsec on the third layer of the OSI model and that we use the PCIe bus to transfer data between the host computer and the FPGA.

**AES Presentation**

The Rijndael AES algorithm was adopted on 26 May 2002 by the NIST to replace the symmetric-key algorithms such as DES or 3DES (Daemen and Rijmen, 2001). The AES algorithm can process data blocks of 128 bits, using cipher keys with lengths of 128, 192 and 256 bits. Both of the keys and the data block are written as matrices, so all the operations are done on matrices.
The state matrix for the data block consists of 4 rows of bytes, each containing $N_b$ bytes, where $N_b$ is the block length divided by a 32-bit word. Thus we obtain $N_b = \text{blocklength}/32 = 4$. Depending on the security requirements, we can choose the key length 128 bits, 192 bits or 256 bits which corresponds to the algorithm name "AES-128", "AES-192" or "AES-256". The AES algorithm consists of a key expansion algorithm that generates the round keys from the primary key. The key length determines the number of rounds $N_r$ in the encryption/decryption process of the AES. The number of rounds for AES-128 is $N_r = 10$ for AES-192 is $N_r = 12$ and for AES-256 is $N_r = 14$. Figure 2 shows the initial round that adds the primary key to the state matrix of the data block. Intermediate rounds from round 1 to round $N_r-1$ that consist of a set of four distinct operations including a substitution operation, a shifting rows and a mixing columns operations and finally, adding a round key operation. The final round $N_r$ does not contain the mix columns operation.

![Figure 2. AES algorithm](image)

**RIFFA 2.0**

RIFFA (Reusable Integration Framework for FPGA Accelerators) is a framework that allows to establish a communication between an FPGA and a host PC using PCIe standard. The host PC can run either on Windows OS or Linux OS (Jacobsen and Kastner, 2013). The RIFFA framework provides a software API with two main functions "data send" and "data receive" that are written in C/C++, Python, Matlab and Java. Those functions are used for sending and receiving data to/from the FPGA connected to the PCIe bus. A hardware interface that allows the transmission/reception of data through a FIFO interface and a DMA (Direct Memory Access). The user needs to interface his hardware architecture directly on the RIFFA framework in order to communicate between the FPGA and the host PC.
The Proposed Solution

The architecture

In order to accelerate the cryptographic transformation for IPsec, we propose the architecture shown on the Figure 3. It consists of two main parts:

a. The host computer comprises the PCIe endpoint, the RIFFA device driver in the Linux Kernel, the RIFFA low level API which provides the software functions "fpga send" and "fpga receive" and finally, the user application software.

b. The FPGA comprises a PCIe endpoint, a RIFFA hardware driver, and the user logic which consists of the implementation of the AES algorithm.

![Figure 3. Architecture proposed to accelerate cryptographic transformation for IPsec](image)

The raw data (the payload) of the IPsec protocol instead of being ciphered in the host PC, are transmitted to the FPGA in order to be ciphered there with an AES algorithm previously implemented in. The data transfer from the host machine to the FPGA is done through the PCIe bus using the RIFFA framework.

Finite state machine of the AES-128 algorithm

The FMS of the AES-128 algorithm shown in the Figure 4 consists of 4 states:

a. Wait for start: It's the initial state of the FSM. When RST=1, the FSM is reset to this state. It waits for the start signal to begin the process.

b. Load Key: Load the key then wait for Ready signal.

c. Load Data: Load the data to encrypt/decrypt and wait for ready signal.

d. Finish process: When the process is finished, we return to the first state.
Figure 4. FSM of the AES-128 algorithm

Finite state machine of the interface between AES-128 algorithm and RIFFA framework

The FSM of the interface between AES-128 and RIFFA hardware driver is shown in the Figure 5. It consists of 6 states:

a. Wait for Rx: It’s the initial state of the FSM. When RST=1, the FSM is reset to this state. It waits for the signal Rx_En to enable data reception.

b. Wait for last data: Reception starts and continues until the last 32 bits word are sent which is indicated by the signal Rx_Last=1. Transition to the next state is then made.

c. AES enable: In this state Start signal is set to 1 in order to start the cipher/decipher operation.

d. Wait result: Wait for the Finish signal to take the value 1 which indicates the end of the cipher/decipher operation.

e. Prepare for Tx: Initialisation of the counter of the words to send before data transmission starts. Transmit data: The end of the transmission is indicated by the signal Tx_Last=1.

Figure 5. FSM of the interface between the AES algorithm and RIFFA framework
Test bench verification for the AES-128 encryption/decryption algorithm

In order to verify the proper functioning of the AES-128 algorithm in encryption/decryption modes, we compare the results with the following FIPS test vector (Dworkin, 2001).

Primary Key: 2b7e151628aed2a6abf71588909cf4c3c
Plain text: 6bc1bee22e409f96e93d7e117393172a
Cipher text: 3ad77bb40d7a3660a89ecaf32466ef97

We can notice in the Figure 6 that the result of the encryption is equal to the cipher text FIPS test vector.

![Figure 6. Test bench of the AES encryption](image)

As well, in the Figure 7 we can notice the result of the decryption is equal to the plain text FIPS test vector.

![Figure 7. Test bench of the AES decryption](image)

The resources utilisation for the implementation of the AES-128 algorithm with the RIFFA framework on a Xilinx Virtex-6 ML605 development board are shown in Table 1. For a clock rate of 250 Mhz, we achieved 391.25 Mbps.

| Table 1. Utilisation for the Implementation of AES Algorithm with RIFFA Framework |
|-----------------------------------|----------------|----------------|
| **Used**                          | Available      | Utilisation    |
| Number of slice registers         | 7395           | 301440         | 2.4%           |
| Number of slice LUTs              | 13146          | 150720         | 8.7%           |
| Number of IOBs                    | 10             | 600            | 1.6%           |
| Number of block RAM/FIFO          | 34             | 52             | 65%            |
| Clock rate                        | 250 Mhz        |                |                |
| Throughput                        |                | 391.25 Mbps    |                |
Conclusion

This paper introduced hardware acceleration of the AES cryptographic in the case of IPsec protocol. The modular structure that characterizes IPsec presents two main advantages. The first one is the offloading of the cryptographic computation from software component to FPGA accelerator through a PCIe bus. The second one is the ease of replacement of the encryption algorithm. The present concept focuses on the AES encryption algorithm in ECB mode which constitutes the most consuming module in terms of computational resources of the IPsec protocol. However, encryption algorithms are usually too complex for high-speed implementations, so the performance of the whole communication chain is usually limited by the encryption subsystem. To avoid overloading the central CPUs of the systems and speed up the communication systems, the security functions are offloaded to FPGA network cards.

References


