IPSec Implementation Project using FPGA and Microcontroller

Guy Gogniat, Wayne Burleson, Mike O'Malley, Lilian Bossuet

IPSec Implementation Project using FPGA and Microcontroller

Guy Gogniat, Wayne Burleson, Mike O'Malley, and Lilian Bossuet

***Abstract*—This paper describes a project that has been performed within the University of Massachusetts, Amherst, USA. The goal of the project was to implement a subset of the IPSec protocol using a PIC microcontroller and an APEX FPGA. The motivation was to enable students to deal with a large scope of skills from network application and cryptography to assembly and VHDL languages and to develop a prototype of their system. Furthermore, as seven students were involved in the project an "industrial" approach was considered through a project manager and several teams.**

In this project many different sub-systems had to communicate with each other to achieve the final product: the PC and the PIC through a serial connection, the PIC and the FPGA through a bidirectional bus, and the PIC and a terminal using a serial connection. Data was to be encrypted within IPSec using an RC6 encryption application.

***Index Terms*—Network application, cryptography, hardware software codesign, IPSec, RC6**

I. Introduction

THE goal of the project was to implement on a PIC microcontroller a subset of the IPSec protocol. IPSec is part of the IPv6 protocol to guarantee the security of data while traveling through the network (i.e. authentication, privacy and integrity). In this project two entities were communicating, a PC and a microcontroller. The PC was sending the data to the microcontroller using a point-to-point protocol over a serial link. Then the microcontroller processed the datagram, checking its validity and extracting the data. In dealing with IPSec, the data was encrypted so it was necessary to first decrypt the data to get the original plain text. Furthermore to speed up the decryption task, a crypto-coprocessor was considered. To manage the project several skills were necessary, from networking to micro-programming and hardware design.

Generally IP is associated with TCP and well known as TCP/IP. In this project in order to manage the complexity of the system the TCP layer was not considered and the data was provided directly after the IP layer as show in Figure 1. Thus from the PC side, the data was encrypted using the RC6 algorithm before being encapsulated into a datagram. To obtain the final datagram two layers were considered which are successively the IP and the PPP layers. The physical layer splits the datagram in order to meet serial link requirements. From the PIC side the same steps were considered but in the reverse order. Once the data was extracted from the datagram it was sent to the crypto-coprocessor in order to retrieve the plain text. The crypto-coprocessor was implemented within an APEX FPGA and connected to the PIC bus.

The project involved seven junior year students from the ECE department of University of Massachusetts, Amherst. They were involved in an honor section project which was an extension on top of 354 and 353 courses where the students already had experience with PIC, and Altera FPGA. They had to move from Verilog to VHDL for this project. Weekly meetings were held in order to discuss the achievements and the difficulties of the project (the project proceeded over eight weeks with final report and demonstration at the end). The goal of the project was to enable students to gather within a same study many skills. From the application side there were security and network considerations, from the architecture side there was HW/SW codesign as the design was split between a PC, a microcontroller and an FPGA. From the programming point of view the students had to deal with C, ASM and VHDL languages. And finally from the prototyping side they had to extend a prototyping board to implement the whole system (refer to Figure 2 for the final demonstrator).

II. Outline of the project

In this project as mentioned previously several aspects were considered, the first one was related to the datagram definition which requires a general understanding of the IP (and IPSec) and PPP layers. The original data was encrypted and gathered with the AH header and the IP header. Each of these headers contains specific information in order to provide a valid datagram. During the project the students have defined the right parameters corresponding to the case study. The resulting datagram was then encapsulated within the PPP layer to provide the final datagram. As for IP, PPP contains specific parameters that were defined. In order to reduce the complexity of the global system a simplified version of the IP and PPP layers was considered (the corresponding protocols

G. Gogniat is with the LESTER Laboratory, University of South Brittany, CNRS FRE 2734, Lorient, France (email: guy.gogniat@univ-ubs.fr)

Wayne Burleson and Mike O'Malley are with the Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA 1003-9284 USA (email: burleson@ecs.umass.edu, momalley@ecs.umass.edu)

Lilian Bossuet is with the IXL laboratory, CNRS UMR5818, ENSEIRB Bordeaux, France (email: bossuet@ixl.fr)

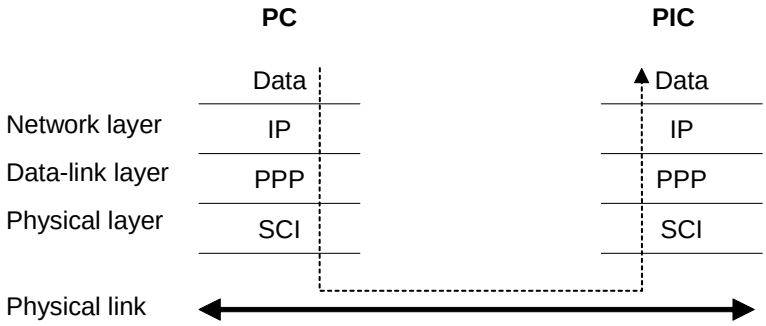

Fig. 1. Layers involved during a communication within the PC and the PIC

can be very complex). For example the SA step was not considered and predefined key and algorithm for the cryptography solution were selected. Furthermore in a first step the authentication algorithm was not handled. Only the cryptography part was targeted. Obviously, the complexity of the system could have evolved depending on the results obtained during the project.

As an initial step, the plan was to manually write in a text file the datagram corresponding to the data to be sent. Then it was necessary to write a program (using the C language) that took this file and transferred it through the serial interface to the PIC microcontroller (Microchip PIC 16F877/874). The microcontroller received the datagram, checked its validity and stored the data in its memory. To provide this functionality it was necessary to configure the serial interface of the PIC in order to be able to receive the datagram. Then the various parameters from the headers were checked to verify the validity of the communication (for example, are the IP source and destination addresses correct). Once that step performed it was necessary to send the data to the crypto-coprocessor to determine the original data (implemented onto an Altera APEX 20KEP20K200 within a Nios Development Kit). All the tasks performed on the microcontroller required quite a large hand-written ASM program, so a rigorous test plan was required for debugging in order to manage the complexity of the code. Finally it was necessary to understand the RC6 cryptographic algorithm to be able to build the corresponding hardware design. For that purpose a VHDL code was defined. In order to help the implementation of the RC6 decryptor an existing design was considered and adapted to the considered case study. Once the data was decrypted it was necessary to send it back to the microcontroller so that it was displayed on a terminal. Figure 3 illustrates the system that has been built and Figure 2 provides the final implementation.

III. Steps of the Project

When dealing with such a project it was important to clearly define the different steps to be reached in order to manage the complexity of the global system and to regularly report progress concerning the state of the project. The following decomposition was considered by the students to build the system (but other solutions were possible).

A. Task 1: PC – PIC communication (it was performed by one student)

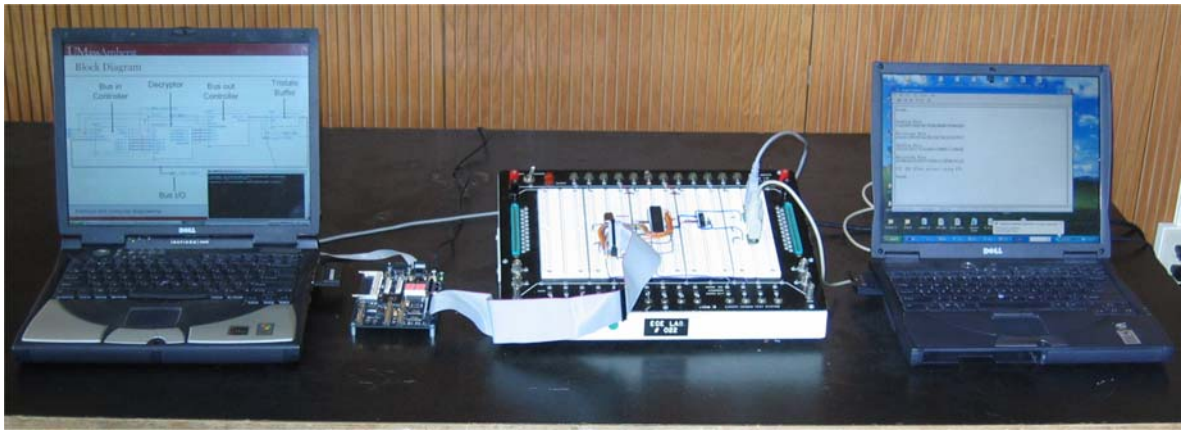

Fig. 2. Implementation of the whole system; the first laptop enables sending the datagram to the PIC; the prototype board contains the PIC and its link with the FPGA; the second laptop is used as a terminal to display the initial datagram and the final plain text

In this part of the project, the task was to write C code which would enable the serial transmission of a file (the datagram), to the PIC and then to write assembly code which would take in the serial transmission and store it into PIC memory. The C code was the first to be tackled for many reasons; the primary reason was the thinking that without any data being sent, there would be no way of testing the PIC code for storing the data into memory. Other reasons for C code being done first was that it was simpler, not requiring any hardware other than a PC and therefore could be done outside of the lab and that it could be done with the least interference with any other parts of the lab.

Serial transmission of the data was accomplished by using the windows.h library, which allowed for com ports to be selected and used to send or receive data at specified baud rates. Once the com port was selected and setup, the rest of the code was simply a matter of opening and preparing the datagram file to be sent through it serially to the PIC.

The assembly was then written for storage of the data sent by the C code. RCIF of the PIR1 register was polled to check for data being received. Once the first byte was detected, it would be stored into PIC memory using indirect addressing and the RCIF would be polled again to check for the next byte to repeat the process. This assembly was later integrated with the other code for the PIC.

B. Task 2: Datagram definition (it was performed by one student)

In a general network stack, data is encapsulated inside of a frame by appending fields to the beginning and end of the data. The datagram represents the final result of data that has been encapsulated by the following process:

1. The original data is wrapped in an IPv6 packet
2. This IPv6 packet is encrypted, which encrypts only the data portion of the packet, and the corresponding Authentication headers are inserted in the packet.
3. This encrypted IPv6 packet is then encapsulated in a PPP frame, which includes data appended to both the beginning and end of the frame.

This complete structure is the final datagram. This structure requires several fields for each step from the above list.

The first step in creating this datagram was to define each of the values for the protocol headers. After determining each constant value and computing the dynamic values, a C program was defined to construct the datagram.

The initial C program was intended to construct a single static datagram in memory, then save that datagram as a file. Starting in this manner provided an easy upgrade path for the planned modification to the code to produce a dynamic datagram based on user input or an input file.

After writing the initial C code to hold the required data, the required functions for the dynamic fields were researched. Specifically, an RC6 encryption algorithm and a CRC checksum function were located and inserted into the code.

C. Task 3: Datagram validation and data extraction (it was performed by two students)

Following the process of storing the datagram in PIC memory, it is required that the PIC checks the validity of certain parameters within the datagram, most importantly the Payload, Destination Address, and Data portions. These parameters are checked to verify the validity of the communication. Following the extraction and checking of each parameter, the next step is to send the encrypted data portion of the datagram to the crypto-processor. The crypto-processor performs decryption and sends the original data (the data before encryption) back to the microcontroller to be stored back in PIC memory. The encrypted data will be written over the decrypted data in the same memory range. Once the data is received from the crypto-processor it is

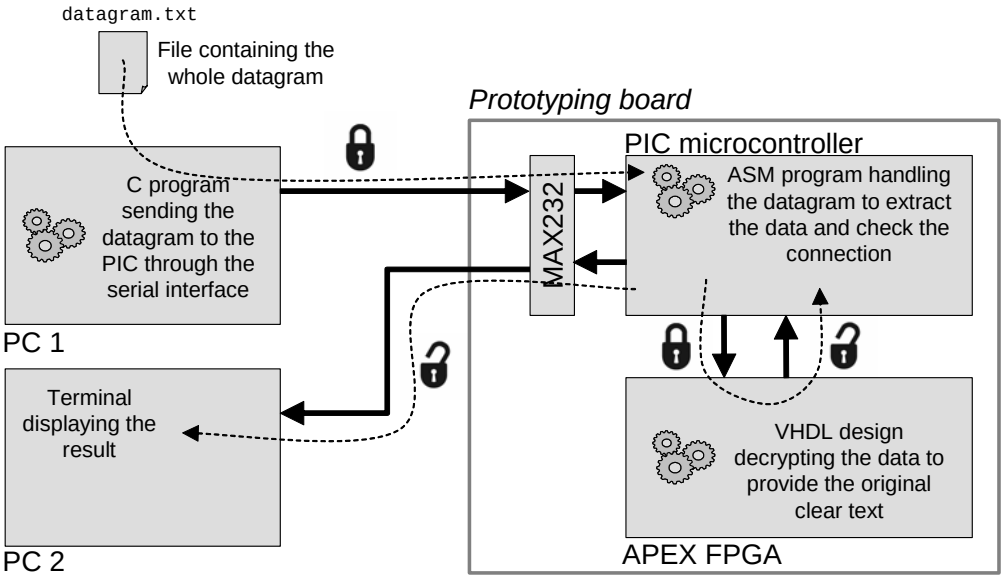

Fig. 3. System partitioning of the IPSec Implementation

displayed on a terminal as shown in Figure 4.

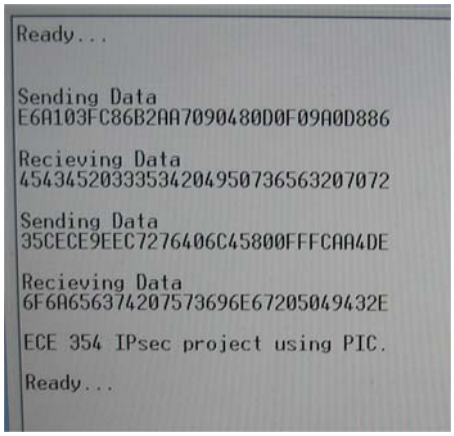

Fig. 4. The terminal displays the cipher text sent to the crypto-processor, the plain text retrieved and the corresponding ASCII text

D. Task 4: Crypto-coprocessor (it was performed by two students)

The first step in the design of the FPGA coprocessor was to define a bus protocol between the PIC and the FPGA. The protocol was to take into account the asynchronous properties of the two devices. In order to implement such a protocol, a hand shake method was used. With the bus protocol decision finalized a high level FSM (Finite State Machine) was designed. The FSM was then split into separate modules: the data input; the decryptor; and the data output.

The beginning of the FPGA consisted of the input (Figure 5). This module was designed to take in series of 16 parallel bits at a time and to output 4 x 32 bits to the decryptor. Each set of 32 bits output to the decryptor consisted of 2 x each 16 originally input, so the input sequence required four buffers, each capable of holding 32 bits.

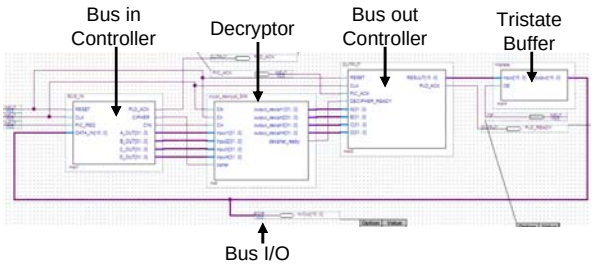

Fig. 5. RTL architecture of the Crypto-processor

The middle of the FPGA code was a decryptor which decrypted 128-bit ciphertext blocks into 128-bit plaintext blocks using the RC6 algorithm. The protocol between the input/output modules and the decryptor had to be established.

The end of the FPGA consisted of the output. The goal of the output was to perform the reverse of the input module. The output took in and stored 4 x 32 bits at once. Each 32-bit word was stored into a 32-bitwide register. Sixteen parallel bits were then sent to the PIC each time the PIC requested data, serially.

E. Task 5: Complete system (it was performed by the project manager + the six previous students)

After each of the four main tasks were tested individually and were successful, the group took on the task of integrating the parts of the system into a whole. This involved a debugging stage. Many bugs were encountered and dealt with. First, the students had to integrate separate pieces of assembly code because three different people wrote portions of it. It was necessary that all of the portions fit and work together as though they were one homogonous piece of code. This involved aligning the variable names, adding code to join two parts, removing parts what were unnecessary or duplicated, and making sure the bus communications matched on both sides. Another design choice used when combining the different parts of the project was being consistent for what banks of memory were used for what purposes.

The integration of the individual tasks into one working project required students setting meeting times so that the whole group could meet or just certain people who were knowledgeable in the current problem area, dividing the labor amongst the group members, prioritizing problems, managing resources, and solving problems as a team. As a result of efficiently meeting and performing the tasks described above, the students were able to produce a working project that met all the goals of the project (Figure 6 provides a detail of the final implementation).

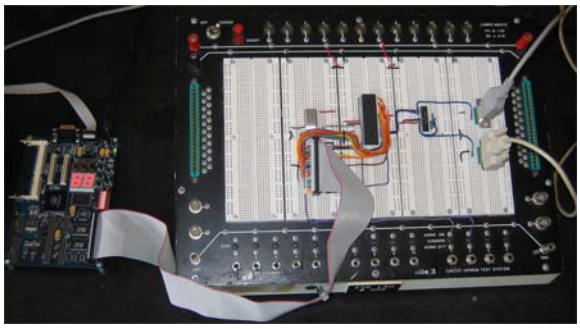

Fig. 6. Prototyping board containing the PIC microcontroller and the links to the emitter, the FPGA and the terminal

IV. FEEDBACK OF THE PROJECT

The project described was very successful. Students praised the project highly for its technical knowledge, organization and management. They enjoyed the large scope of the required skills. Some of the comments are: "it was a good time working in a large group", "the project gave me organization skills and communication skills", "I was very impressed with the outcome of the project", "the project was very well-rounded, it contained C code, VHDL code and PIC assembly code".

V. Conclusion

This paper presents a project that has been carried out within the University of Massachusetts, Amherst, USA dealing with an IPSec design through an interdisciplinary approach. Students had the opportunity to use skills from several courses within a single project and to work in a large group. They had to handle networking, processor based design and reconfigurable architecture which provide a good overview of a system level design.

Seven students have worked on the project (which was good for a first experience) but it could have been possible to increase the number of students as several parts like authentication or key management were not considered. Another nice extension of the project could be to dynamically adapt the cryptography algorithms in order to take benefit of dynamic reconfiguration.

This year the experience is carried out and extended within the ENSEIRB engineering school in France.

Acknowledgment

The authors would like to thanks Professor Tilman Wolf and Hieng Shea, Jared Eldridge, Ben Lapointe, Matt Brennan, Brian Roberts, Dan Verdolino who have been involved in the honor section project of the ECE 354 course within the ECE department of University of Massachusetts, Amherst.

References

[1] Internetworking Technology Handbook. [Online]. Available: http: //www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/

[2] IP Authentication Header. [Online]. Available: http://www.cse.ohio-state.edu/cgi-bin/rfc/rfc2402.html#sec-2

[3] Point-to-Point Protocol. [Online]. Available: http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/ppp.htm

[4] RC6 implementation. [Online]. Available: http://perso.ens-lyon.fr/jean-luc.beuchat/RC6/#ref_rc6

[5] R.L. Rivest, M. J. B. Robshaw, R. Sidney, and Y. L. Yin. The RC6 Block Cipher. In First Advanced Encryption Standard (AES) Conference, 1998.

[6] Jean-Luc Beuchat. FPGA Implementations of the RC6 Block Cipher. In P. Y. K. Cheung, G. A. Constantinides, and J. T. de Sousa, editors, Field-Programmable Logic and Applications, number 2778 in Lecture Notes in Computer Science, pages 101-110. Springer, 2003.