Xây dựng giải pháp trao đổi khóa IKEv2 sử dụng NIOS II trên FPGA

Vũ Tá Cường, La Hữu Phúc

***Tóm tắt*—Giao thức Internet Key Exchange (IKE) là một giao thức thực hiện quá trình trao đổi khóa và thỏa thuận trong chế độ bảo mật IPSec. Để thực thi giao thức bảo mật IPSec tốc độ cao thì thường kết hợp giữa phần mềm và phần cứng trên vi mạch Field Programmable Gate Array (FPGA) [7], [8]. Trong đó, các thao tác mật mã, đóng gói và bóc tách gói tin được thực hiện bằng FPGA để đảm bảo thực hiện hệ thống IPSec tốc độ cao; giao thức trao đổi khóa IKE được thực hiện bằng phần mềm sử dụng hệ điều hành Linux nhúng. Trong bài báo này, nhóm tác giả giới thiệu giải pháp thực hiện giải thuật trao đổi khóa IKE sử dụng Nios II trên FPGA. Với cách tiếp cận này, nhóm tác giả đã tự tổ chức, xây dựng chương trình trên bộ vi xử lý, nhờ đó kiểm soát được toàn bộ dòng dữ liệu.**

***Abstract*—IKE (Internet Key Exchange) is a protocol that performs key exchange and agreement process in IPSec security mode. To implement high speed IPSec security protocol, it is often combined software and hardware on Field Programmable Gate Array (FPGA) [7], [8]. Therein, encryption, packet encapsulation and extraction operations will be performed by FPGA to ensure high speed IPSec system implementation; the IKE protocol is implemented by software using an embed Linux operating system. In this paper, the authors introduce the solution of implementing IKE key exchange algorithm using Nios II on FPGA. With this approach, the authors have organized and built the program on the microprocessor by themselves, therefore the entire data stream is controlled.**

Từ khóa—IPSec; IKE; FPGA, Nios II.

Keyword—IPSec; IKE; FPGA, Nios II.

I. GIỚI THIỆU

Trong những năm gần đây, các cuộc tấn công mạng xảy ra ngày càng phổ biến, tính chất của các vụ tấn công ngày càng phức tạp và khó lường. Có rất nhiều các giải pháp công nghệ, sản phẩm đã ra đời nhằm hạn chế các rủi ro nêu trên, trong đó phải kể đến các sản phẩm bảo mật IPSec. Các sản phẩm này thường chia thành hai nhóm chính: sản phẩm bằng phần mềm và sản phẩm bằng phần cứng. Các sản phẩm bằng phần mềm thường được cài đặt trong các hệ điều hành có ưu điểm là mềm dẻo, dễ cài đặt, dễ nâng cấp và tùy biến. Tuy nhiên, tốc độ và độ an toàn trong nhiều trường hợp còn hạn chế. Chính vì thế, trong nhiều hệ thống đòi hỏi băng thông lớn, tốc độ đường truyền cao, người ta thường sử dụng các sản phẩm bằng phần cứng.

Hiện nay, các sản phẩm bảo mật IPSec thường được thiết kế dựa trên vi mạch FPGA, vì nó cho phép chế tạo ra các thiết bị có tốc độ mã hóa và độ bảo mật cao. Trong đó, phần mật mã sẽ được thực hiện bằng ngôn ngữ mô tả phần cứng, giao thức trao đổi khóa sẽ được thực hiện bằng phần mềm [7], [8]. Trong các sản phẩm này, giao thức trao đổi khóa IKE được thực hiện trên hệ điều hành Linux nhúng. Cách tiếp cận này có ưu điểm là dễ cài đặt do tận dụng được bộ IP/TCP stack có sẵn, cũng như các bộ phần mềm trao đổi khóa IKE mã nguồn mở. Tuy nhiên, do sử dụng hệ điều hành nên các tác vụ xử lý đều do hệ điều hành quản lý, nên khó kiểm soát được toàn bộ dòng dữ liệu, dễ gây lộ lọt thông tin.

Vì vậy, giải pháp thực hiện trao đổi khóa IKE sử dụng lõi bộ vi xử lý nhúng trên FPGA được cho là an toàn hơn. Tuy nhiên, khó khăn lớn nhất của giải pháp này là cần tự tổ chức và xây dựng toàn bộ các giao thức xử lý dữ liệu gói tin, các giao thức truyền thông, các bộ thư viện tính toán số lớn phục vụ quá trình trao đổi khóa IKE. Trong bài báo này, nhóm tác giả giới thiệu cách thực hiện giải thuật trao đổi khoá IKEv2 (theo RFC 7296 [6]) sử dụng bộ vi xử lý Nios II trên FPGA.

Bài báo được nhận ngày 25/3/2021. Bài báo được nhận xét bởi phản biện thứ nhất ngày 26/5/2021 và được chấp nhận đăng ngày 10/6/2021. Bài báo được nhận xét bởi phản biện thứ hai ngày 02/6/2021 và được chấp nhận đăng ngày 18/6/2021.

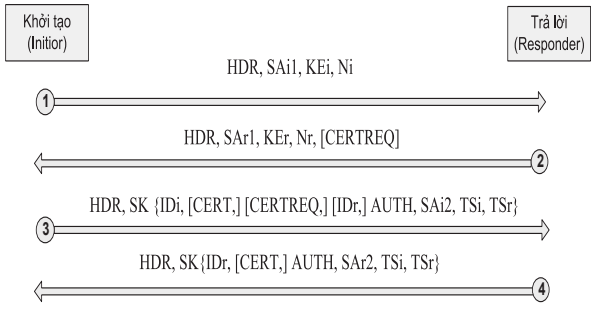
II. ĐỀ XUẤT GIẢI PHÁP THỰC HIỆN TRAO ĐỔI KHÓA IKE SỬ DỤNG NIOS II

A. Tổng quan về giao thức IKE

IKE thực hiện xác thực 2 chiều và thiết lập một tổ hợp an toàn IKE (IKE security association – IKE SA) giữa 2 thực thể, sau đó IKE SA được sử dụng để thiết lập các tổ hợp an toàn cho giao thức các ESP (Encapsulating Security Payload) hoặc AH (Authentication Header). Ta gọi tổ hợp an toàn dùng cho IKE là "IKE_SA". Các tổ hợp an toàn sử dụng cho ESP/AH (được thiết lập thông qua IKE_SA) ta gọi là "CHILD_SA" hoặc "IPSEC_SA".

Mọi truyền thông trong IKE gồm các cặp thông báo: một thông báo yêu cầu (request) và một thông báo trả lời (response). Cặp thông báo này gọi là một "trao đổi" (exchange). Trong IKEv2 có các kiểu trao đổi là: IKE_SA_INIT, IKE_AUTH, CREATE_CHILD_SA và INFORMATION. Ta gọi các thông báo đầu tiên để thiết lập một IKE_SA là trao đổi IKE_SA_INIT và IKE_AUTH. Nhìn chung, sẽ có một trao đổi IKE_SA_INIT và một trao đổi IKE_AUTH để tạo ra một IKE_SA và CHILD_SA đầu tiên (tổng là 4 thông báo). Trong mọi trường hợp, tất cả các trao đổi IKE_SA_INIT phải hoàn thành đầu tiên, sau đó các trao đổi IKE_AUTH phải hoàn thành và tiếp theo là bất kỳ trao đổi CREATE_CHILD_SA hoặc INFORMATION.

Sơ đồ hoạt động của pha trao đổi IKE được thể hiện cụ thể như trong Hình 1.



Hình 1. Sơ đồ hoạt động của IKE.

Trong đó, cặp thông báo 1 và 2 gọi là trao đổi IKE_SA_INIT, cặp thông báo 3 và 4 gọi là trao đổi IKE_AUTH.

Bước 1: Bên khởi tạo gửi một thông báo yêu cầu IKE_SA_INIT gồm các tham số:

- *HDR*: gồm chỉ số tham số an toàn (SPI – Security Parameter Index), số phiên bản, và các cờ khác;
- *SAi1*: gồm các thuật toán mật mã mà bên khởi tạo có thể hỗ trợ cho tổ hợp IKE_SA;
- *KEi*: Giá trị DH của bên khởi tạo;
- *Ni*: Số ngẫu nhiên nonce của bên khởi tạo.

Bước 2: Bên trả lời chọn một bộ thuật toán mật mã từ danh sách các thuật toán mật mã mà bên khởi tạo đưa ra, và chứa sự lựa chọn đó trong phần nội dung *SAr*1; [CERTREQ] là lựa chọn gửi yêu cậu chứng chỉ hoặc không (danh sách các CA mà bên trả lời hỗ trợ); các thành phần tương ứng còn lại như đối với bên khởi tạo.

Khi hoàn thành trao đổi IKE_SA_INIT, mỗi bên có thể sinh ra SKEYSEED, từ đó tất cả các khóa sẽ được sinh ra cho IKE_SA. Mọi phần nội dung từ phần đầu của tất cả các thông báo tiếp theo sẽ được mã hóa và được bảo vệ tính toàn vẹn. Các khóa được dùng cho mã hóa và bảo vệ tính toàn vẹn được tạo ra từ SKEYSEED như sau:

- *SK_e*: khóa mã hóa;
- *SK_a*: khóa xác thực và bảo vệ tính toàn vẹn.

Mỗi chiều truyền thông sử dụng một *SK_e* và *SK_a* riêng biệt. Ngoài khóa *SK_e* và *SK_a* được tạo ra từ giá trị DH cho IKE_SA, thì một khóa *SK_d* cũng được tạo ra và được sử dụng để tạo các khóa khác cho CHILD_SA. Kí hiệu *SK*{...} chỉ ra rằng những phần nội dung trong đó sẽ được mã hóa và bảo vệ tính toàn vẹn bằng các khóa *SK_e* và *SK_a* cho mỗi chiều truyền thông.

Bước 3: Bên khởi tạo gửi yêu cầu IKE_AUTH gồm các tham số:

- *HDR*: IKE header;
- *IDi*: Định danh bên khởi tạo;
- *[CERT]*: Chứng chỉ (nếu được yêu cầu);
- *[CERTREQ]*: Yêu cầu chứng chỉ (tuỳ chọn);

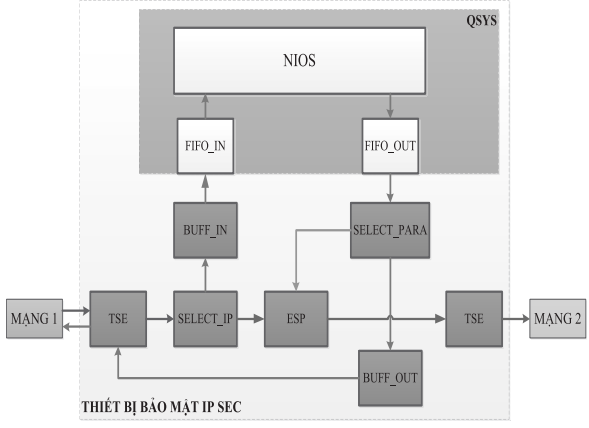
- *AUTH*: Bảo vệ tính toàn vẹn của thông báo Bước 1;
- *[IDr]*: Định danh bên trả lời (tuỳ chọn);
- *SAi2*: Các thuật toán mà bên khởi tạo có thể hỗ trợ cho IPSEC_SA;
- *TSi*, *TSr*: Thoả thuận truyền thông cần được bảo vệ.

Bước 4: Bên trả lời gửi lại phản hồi IKE_AUTH gồm các tham số:

- *HDR*: IKE header;
- *IDr*: Định danh bên trả lời;
- *[CERT]*: Chứng chỉ (nếu được yêu cầu);
- *AUTH*: Bảo vệ tính toàn vẹn của thông báo Bước 1;
- *SAr2*: Bộ các thuật toán hỗ trợ IPSEC_SA có trong danh sách các thuật toán mà bên khởi tạo đề xuất (trong *SAi2*);
- *TSi*, *TSr*: Thoả thuận truyền thông cần được bảo vệ.

B. Đề xuất giải pháp thực hiện trao đổi khóa IKE sử dụng Nios II trên FPGA

Mô hình thiết bị bảo mật IPSec sử dụng FPGA được thiết kế như trong Hình 2.



Hình 2. Mô hình thiết bị bảo mật IPSec sử dụng FPGA.

Trong đó, phần FPGA sẽ làm nhiệm vụ bóc tách, xử lý gói tin bằng ngôn ngữ mô tả phần cứng HDL. Bộ vi xử lý Nios sẽ làm nhiệm vụ quản lý trao đổi khóa IKE và cấu hình thiết bị. Giải pháp trao đổi khóa IKE sử dụng Nios II trên FPGA được nhóm tác giả thiết kế như sau: khối TSE sẽ đọc ghi gói tin từ cổng mạng đưa vào thiết bị để sẵn sàng xử lý. Dữ liệu sau khối TSE được đưa vào khối SELECT_IP, khối này có nhiệm vụ phân tích gói tin nhằm xác định các gói tin là ESP hay IKE. Các gói IKE sẽ được đưa đến khối BUFF_IN trước khi gửi đến FIFO_IN để Nios đọc và xử lý gói IKE phục vụ việc trao đổi khóa. Bộ vi xử lý Nios thực hiện xử lý trao đổi khóa IKE, gửi gói tin IKE xuống FPGA qua FIFO_OUT. Kết thúc quá trình trao đổi khóa, Nios gửi khóa ESP xuống FPGA qua FIFO_OUT. Khối SELECT_PARA đọc dữ liệu do Nios gửi xuống từ FIFO_OUT, phân tích gói tin nhằm xác định gói tin IKE và gói tin chứa khóa ESP. Nếu là gói tin IKE sẽ được gửi ra BUFF_OUT để chuyển sang khối TSE đóng gói và gửi đi. Nếu là gói tin chứa khóa ESP sẽ được gửi sang khối ESP.

Như vậy giải pháp của nhóm nghiên cứu ở đây là các gói tin ESP sẽ được xử lý trực tiếp dưới FPGA bằng ngôn ngữ mô tả phần cứng (Hardware Description Language - HDL), còn gói tin IKE sẽ được FPGA gửi lên Nios để xử lý. Trong bài báo này, nhóm tác giả không trình bày giải pháp xử lý gói tin ESP (mã hóa dữ liệu), mã sẽ tập trung trình bày giải pháp thực hiện IKEv2 sử dụng bộ vi xử lý Nios. Việc xử lý gói tin IKE là phức tạp, nếu thực hiện trên FPGA sẽ gặp nhiều khó khăn. Ngoài ra, hoạt động trao đổi khóa chỉ diễn ra ở đầu phiên liên lạc, hoặc khi có yêu cầu trao đổi khóa lại, do đó việc thực hiện trao đổi khóa trên Nios vẫn đủ để đáp ứng tốc độ cho thiết bị bảo mật IPSec.

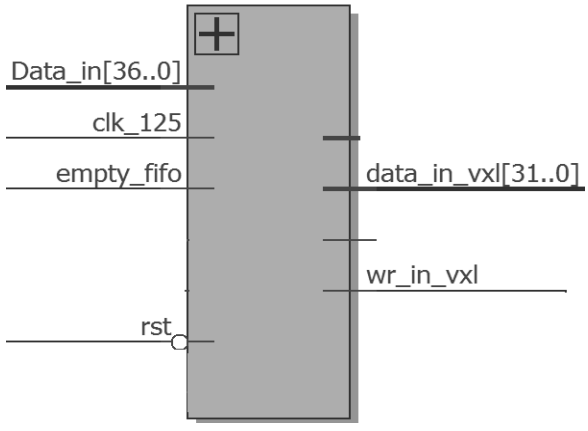
III. GIẢI PHÁP TRAO ĐỔI DỮ LIỆU GIỮA NIOS VÀ FPGA

A. Giải pháp trao đổi dữ liệu với Nios trên FPGA

Theo sơ đồ trong Hình 2, các gói tin nhận được từ khối TSE sẽ được kiểm tra bởi khối SELECT_IP. Theo RFC 7296 [6] giao thức IKEv2 chạy trên UDP port 500 hoặc 4500, do đó các gói tin IKE (đóng theo cấu trúc gói tin UDP có Type protocol là 0x11) sẽ được phân luồng lên cho NIOS để tạo ra các tham số phục vụ cho quá trình xử lý gói tin ESP. Việc sử dụng UDP cho IKEv2 sẽ giúp tương thích ngược với IKEv1. Hiện nay, theo RFC 8229 [10] có thể sử dụng tính năng đóng gói TCP cho

IKEv2. Tuy nhiên trong phần 12 của RFC 8229 có chỉ rõ là việc sử dụng như vậy có một số tác động tiêu cực đến hiệu suất. Do đó, việc sử dụng UDP cho IKEv2 thường được ưu tiên hơn. Ngoài ra, đây là tiêu chuẩn mở rộng mới được ban hành, nên chưa phổ biến.

Các gói tin IKE sau khi được phân luồng để truyền lên cho Nios sẽ được đẩy vào trong khối BUFFER_IN như trên Hình 2. Sơ đồ chân tín hiệu của khối BUFFER_IN được thể hiện như trên Hình 3.



Hình 3. Cấu trúc khối BUFFER_IN.

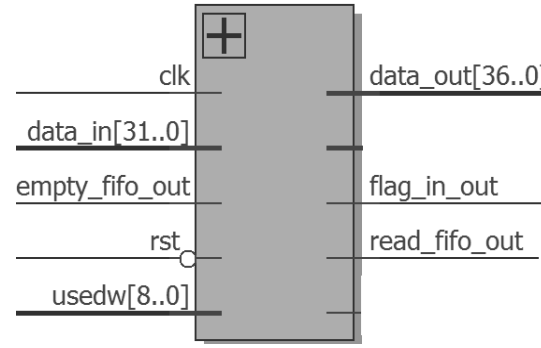
Đầu vào:

- *Data_in*: 36 bit dữ liệu theo cấu trúc ST-Avalon (st_mod[1:0], st_eop, st_sop, st_vlid, st_data[31:0]).
- *empty_fifo*: tín hiệu báo dữ liệu trong bộ nhớ FIFO_IN. Khi có dữ liệu trong FIFO, tín hiệu này sẽ hạ xuống mức thấp (1'b0). Tín hiệu này dùng để làm tín hiệu điều khiển quá trình ghi dữ liệu từ khối BUFFER_IN vào cho FIFO.

Đầu ra:

- *wr_in_vxl*: tín hiệu điều khiển quá trình ghi dữ liệu vào trong bộ nhớ FIFO_IN.
- *data_in_vxl*: dữ liệu của gói tin được đẩy lên cho Nios.

Các gói tin IKE và các bộ tham số mật mã sau khi đã thực hiện trao đổi khóa xong sẽ được bộ vi xử lý ghi vào bộ nhớ FIFO_OUT. Dưới FPGA sẽ thực hiện quá trình đọc dữ liệu từ FIFO và được xử lý bởi khối SELECT_PARA. Sơ đồ của khối được thể hiện như trên Hình 4.



Hình 4. Sơ đồ khối SELECT_PARA.

Đầu vào:

- *data_in*: dữ liệu gói tin IKE hoặc bộ tham số mật mã.
- *empty_fifo_out*: tín hiệu báo hiệu dữ liệu trong FIFO_OUT. Nếu trong FIFO_OUT có dữ liệu, tín hiệu này sẽ bật về mức thấp (1'b0).
- *usedw*: Số từ (32 bit) có trong FIFO.

Đầu ra:

- *data_out*: 36 bit, được đóng gói theo cấu trúc của bus ST-Avalon.
- *flag_in_out*: tín hiệu báo hiệu cho các khối ESP và BUFFER_OUT biết gói tin là IKE hay gói chứa bộ tham số mật mã.
- *read_fifo_out*: tín hiệu cho phép đọc dữ liệu ra từ FIFO_OUT.

Khối SELECT_PARA hoạt động dựa trên 32 bit điều khiển được chèn vào phía đầu tiên của mỗi gói tin IKE hay bộ tham số được NIOS đẩy xuống.

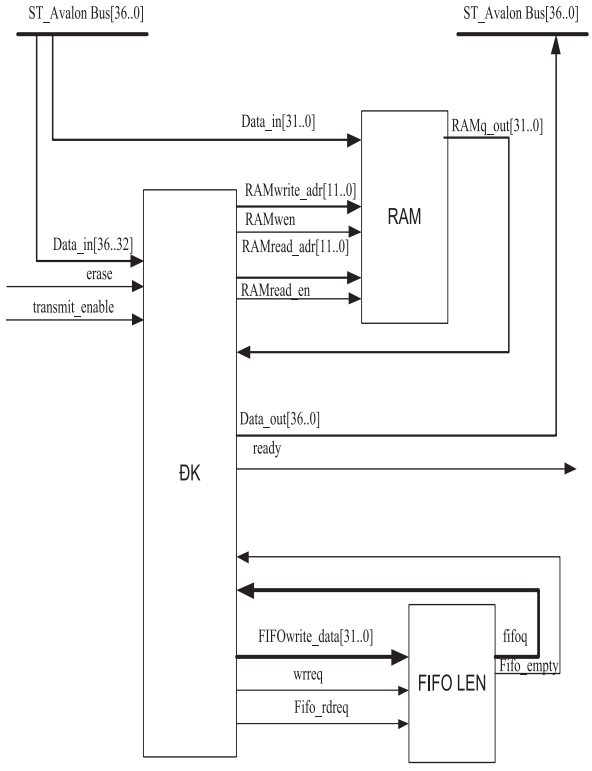
Để thực hiện xử lý gói tin trên FPGA, nhóm tác giả sử dụng bộ nhớ RAM và bộ nhớ FIFO được thiết kế như trên Hình 5.

Gói tin được truyền từ FPGA lên cho Nios hay ngược lại sẽ được lưu vào bộ nhớ RAM, còn bộ nhớ FIFO dùng để lưu kích thước của từng gói tin (được tính theo số từ 32 bit).

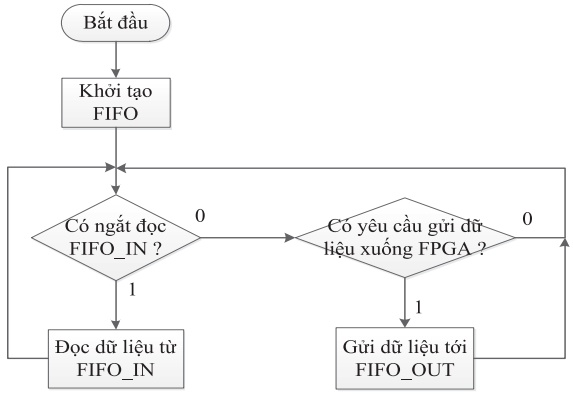
B. Giải pháp trao đổi dữ liệu với FPGA trên Nios

Giải pháp trao đổi dữ liệu giữa FPGA và Nios được thực hiện thông qua FIFO_IN và FIFO_OUT như trong Hình 2. Nios II điều khiển quá trình đọc dữ liệu từ FIFO_IN theo ngắt, và

điều khiển truyền dữ liệu xuống FPGA thông qua FIFO_OUT. Lưu đồ hoạt động của bộ vi xử lý Nios được tổ chức như trong Hình 6.



Hình 5. Giải pháp xử lý gói tin trên FPGA.



Hình 6. Lưu đồ hoạt động trao đổi dữ liệu giữa FPGA của Nios II.

Hoạt động trao đổi dữ liệu với FPGA của bộ vi xử lý được mô tả như sau: Khi thiết bị khởi động, bộ vi xử lý sẽ tiến hành khởi tạo FIFO, cài đặt các tham số cho FIFO để phục vụ việc trao đổi dữ liệu giữa bộ vi xử lý và FPGA. Sau khi khởi tạo và cài đặt xong, bộ vi xử lý ở trạng thái chờ chỉ lệnh. Nếu có ngắt đọc từ FIFO_IN, bộ vi xử lý sẽ tiến hành đọc dữ liệu từ FIFO_IN sau đó quay lại trạng thái chờ chỉ lệnh. Nếu có yêu cầu gửi dữ liệu xuống FPGA, bộ vi xử lý sẽ gửi dữ liệu tới FIFO_OUT sau đó quay lại trạng thái chờ chỉ lệnh. Hoạt động của bộ vi xử lý sẽ là vòng lặp vô hạn, đảm bảo việc trao đổi dữ liệu giữa bộ vi xử lý và FPGA được diễn ra theo đúng yêu cầu của bài toán.

Để việc trao đổi dữ liệu giữa Nios và FPGA hoạt động đúng theo yêu cầu đặt ra, nhóm tác giả tổ chức cấu trúc gói tin Nios gửi xuống FPGA với header 32 bit được thêm vào dữ liệu cần gửi xuống FPGA. Trong header, nhóm tác giả sử dụng 16 bit để định dạng gói tin gửi từ Nios xuống FPGA, cụ thể như sau:

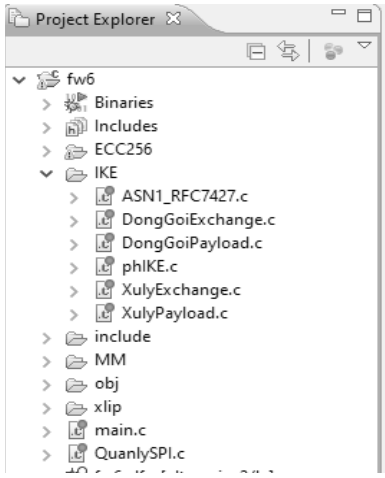
15	14	13..12	11..9	8..0
type	direction	mod	address	len

- len: độ dài gói tin cần gửi (tính theo DWORD 4 byte)
- address: địa chỉ lưu trong RAM
- mod: chỉ ra số bit của nghĩa trong 32 bit cuối cùng của gói tin
  - 11: data[23:0] không có nghĩa
  - 10: data[15:0] không có nghĩa
  - 01: data[7:0] không có nghĩa
  - 00: data[31:0] có nghĩa
- direction: hướng truyền gói tin
  - 0: cổng 0
  - 1: cổng 1
- type: loại gói tin
  - 0: gói tin IKE
  - 1: gói tin chứa khóa ESP

IV. GIẢI PHÁP THỰC HIỆN TRAO ĐỔI KHÓA IKE TRÊN NIOS II

Cấu trúc chương trình trên Nios được tổ chức như trong Hình 7, trong đó mã nguồn được chia ra làm các thư mục theo chức năng. Mã nguồn phục vụ tính toán ECC sẽ đặt trong thư mục "ECC" mã nguồn phục vụ xử lý trao đổi khóa sẽ đặt trong thư mục "IKE"; mã nguồn phục vụ tính toán mật mã sẽ đặt trong thư mục "MM"; mã nguồn phục vụ xử lý các gói tin, trao đổi dữ liệu giữa Nios và FPGA sẽ đặt trong thư mục "XLIP"; mã nguồn sử dụng trong giao thức

ARP sẽ đặt trong file `QuanlySPI.c`; mã nguồn hàm chính sẽ đặt trong file `main.c`.



Hình 7. Cấu trúc mã nguồn Nios II.

Trong đó, mã nguồn phục vụ xử lý trao đổi khóa IKE được tổ chức theo 3 lớp:

- Lớp 1: chứa mã nguồn xử lý và đóng gói payload (file `XulyPayload.c` và `DongGoiPayload.c`).
- Lớp 2: chứa mã nguồn xử lý và đóng gói các gói tin trao đổi IKE (file `XulyExchange.c` và `DongGoiExchange.c`).
- Lớp 3: chứa mã nguồn xử lý trao đổi IKE (file `phIKE.c`).

Để nhận dữ liệu từ FPGA gửi qua FIFO_IN, nhóm tác giả tổ chức lưu dữ liệu nhận được vào mảng trước khi xử lý, với kích thước cho phép nhận tối đa 4 gói tin.

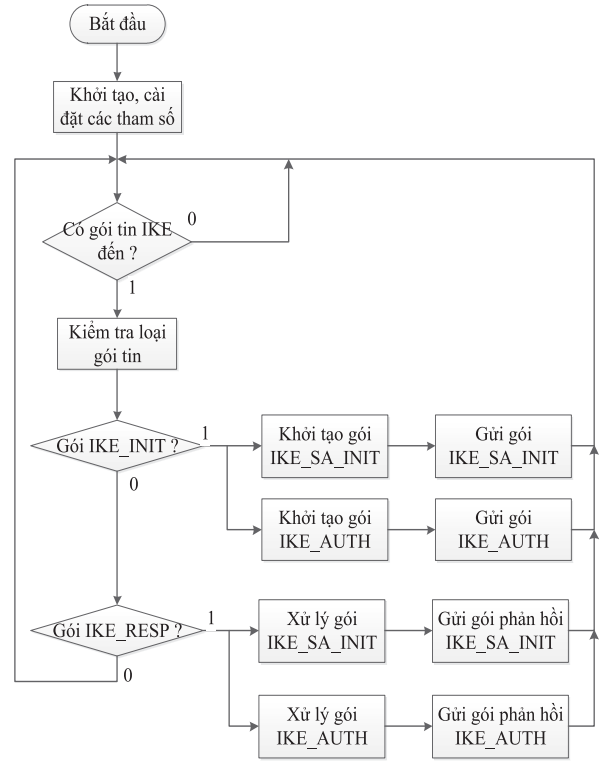
```
unsigned char pkt_data[4][BUFFER_LEN];
```

Việc tổ chức lưu dữ liệu nhận được trước khi xử lý đảm bảo không bị mất dữ liệu khi Nios xử lý gói tin. Số gói tin tối đa cho phép nhận cũng có thể tăng lên (lớn hơn 4), tuy nhiên điều đó sẽ làm tiêu tốn tài nguyên của chương trình. Do việc xử lý gói tin được tiến hành song song với việc nhận gói tin từ FIFO, vì vậy khi có gói tin trong bộ đệm nhận dữ liệu `pkt_data` chương trình sẽ tiến hành xử lý ngay để giải phóng bộ đệm nhận. Thông thường khi gói tin kế tiếp được nhận từ FIFO, thì gói tin trước đó đã được firmware Nios xử lý xong. Qua thử nghiệm thực tế, nhóm tác giả nhận thấy số gói tin tối đa cho phép nhận là 4 đáp ứng được việc nhận gói tin từ FIFO_IN, không xảy ra hiện tượng mất gói cũng như tràn bộ đệm nhận `pkt_data`. Khi có ngắt đọc dữ liệu từ FIFO_IN, Nios sẽ đọc toàn bộ gói tin FPGA gửi qua FIFO_IN và lưu vào mảng `pkt_data`.

Trong chương trình chính, nhóm tác giả tổ chức một vòng lặp vô hạn đọc dữ liệu từ mảng `pkt_data` để xử lý như sau:

```
while (1){
    dodaigoinhan    =    ReadPKT(goinhan,
header);
    if (dodaigoinhan!=-1) {
    //tien hanh xu ly goi tin IKE
    ...
    }
}
```

Lưu đồ hoạt động chung của chương trình xử lý trao đổi khóa IKE sử dụng Nios được tổ chức như trong Hình 8.



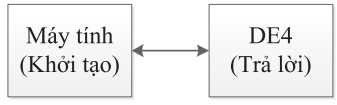
Hình 8. Lưu đồ hoạt động chung của chương trình trao đổi khóa IKE sử dụng Nios.

Khi nhận được gói tin IKE, bộ vi xử lý Nios sẽ tiến hành xử lý gói tin, tạo ra các gói IKE phản hồi gửi xuống FPGA qua FIFO_OUT để phục vụ quá trình trao đổi khóa. Kết thúc quá trình trao đổi khóa, khóa ESP được sinh ra sẽ gửi xuống FPGA.

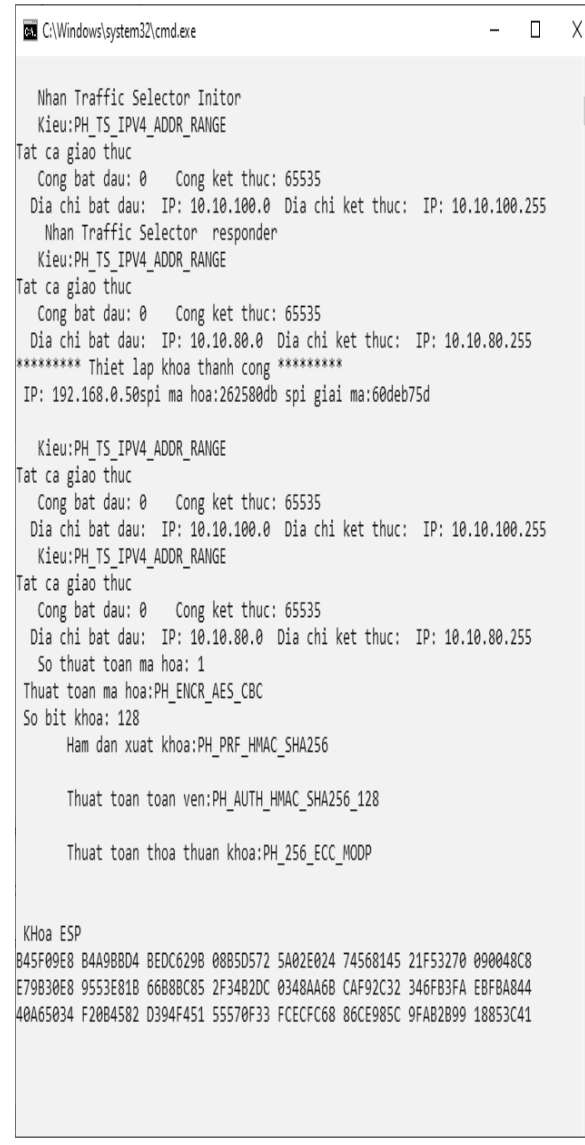
V. KẾT QUẢ

Như đã trình bày, các phương pháp khác thường thực thi IKE trên hệ điều hành Linux nhúng, do đó tận dụng được bộ IP/TCP stack có sẵn, cũng như các bộ phần mềm trao đổi khóa IKE mã nguồn mở. Với cách tiếp cận thực hiện IKE trên Nios, nhóm nghiên cứu sẽ cần tự tổ chức và xây dựng toàn bộ các giao thức xử lý dữ liệu gói tin, các giao thức truyền thông và giao thức IKE đảm bảo hoạt động đúng theo tiêu chuẩn IKEv2 trong RFC 7296 [6].

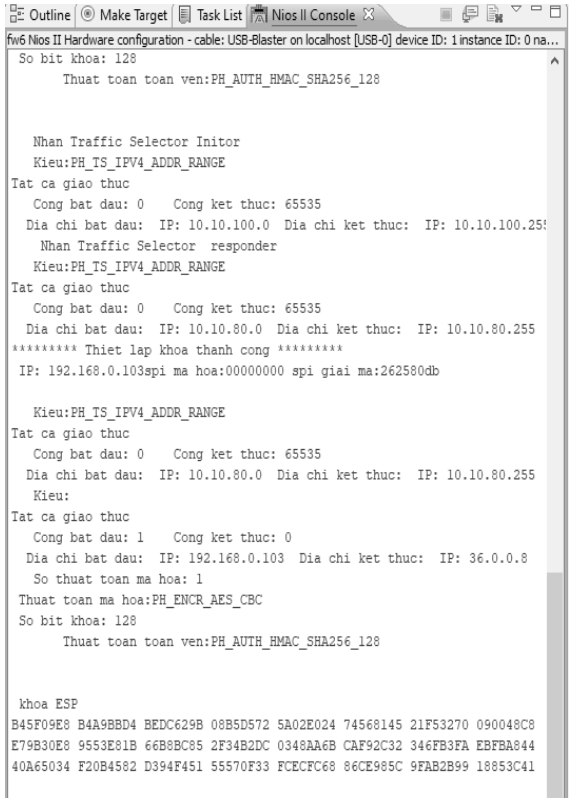
Vì vậy, sau khi xây dựng thành công giải pháp thực hiện trao đổi khóa IKE sử dụng Nios II trên FPGA, nhóm tác giả đã tiến hành thử nghiệm trên bo mạch phát triển DE4. Nhóm tác giả sử dụng một máy tính kết nối với bo mạch DE4 để thực hiện trao đổi khóa IKE giữa máy tính và bo mạch DE4 (Hình 9). Trong đó, máy tính sẽ đóng vai trò là bên khởi tạo, còn bo mạch DE4 sẽ đóng vai trò là bên trả lời. Trên máy tính sẽ cài đặt bộ phần mềm trao đổi khóa mã nguồn mở strongSwan [9], trên thiết bị DE4 sẽ cài đặt firmware trao đổi khóa IKEv2 sử dụng Nios do nhóm tác giả tự phát triển. Kịch bản thử nghiệm này sẽ giúp khẳng định firmware trao đổi khóa IKEv2 sử dụng Nios do nhóm tác giả tự phát triển hoạt động đúng theo chuẩn trong RFC 7296 [6].



Hình 9. Mô hình thử nghiệm giải pháp trao đổi khóa IKE.



Hình 10. Kết quả trao đổi khóa của chương trình máy tính.



Hình 11. Kết quả trao đổi khóa của chương trình Nios II.

Kết quả của quá trình trao đổi khóa giữa máy tính và bo mạch DE4 được thể hiện như trong Hình 10 và Hình 11.

Ngoài ra nhóm cũng tiến hành thử nghiệm để đánh giá tốc độ trao đổi khóa giữa máy tính và bo mạch DE4. Kết quả cụ thể được thể hiện trong bảng sau.

BẢNG 1. KẾT QUẢ THỬ NGHIỆM TRAO ĐỔI KHÓA

Lần thứ	Kết quả	Thời gian (giây)
1	Thành công	2,39
2	Thành công	2,42
3	Thành công	2,56
4	Thành công	2,35
5	Thành công	2,44
6	Thành công	2,33
7	Thành công	2,45
8	Thành công	2,50
9	Thành công	2,37
10	Thành công	2,48

Kết quả thử nghiệm trên cho thấy quá trình trao đổi khóa giữa máy tính và bo mạch DE4 hoạt động ổn định, đạt được yêu cầu đặt ra, thời gian trao đổi khóa trung bình dưới 3 giây.

Trong các nghiên cứu liên quan [7, 8], các tác giả thực thi IKEv2 bằng cách sử dụng bộ mã nguồn mở và cấu hình lại các tham số cho phù hợp với yêu cầu bài toán. Do đó, với giải pháp tự tổ chức và thực hiện IKEv2 sử dụng Nios của nhóm tác giả sẽ cho phép kiểm soát toàn bộ luồng dữ liệu, can thiệp sâu vào IKEv2 mà vẫn đảm bảo thiết bị hoạt động theo đúng chuẩn quốc tế.

KẾT LUẬN

Bài báo đã trình bày việc nghiên cứu xây dựng và tích hợp giải pháp trao đổi khóa IKE sử dụng Nios II trên FPGA. Nhóm tác giả đã nghiên cứu đưa ra giải pháp trao đổi dữ liệu giữa FPGA và Nios, cũng như tổ chức thành công chương trình trao đổi khóa IKE sử dụng Nios II. Toàn bộ mã nguồn trao đổi khóa IKE sử dụng Nios do nhóm tác giả tự tổ chức và xây dựng, vì thế kiểm soát hoàn toàn quá trình xử lý dữ liệu, tránh phát sinh các lỗ hổng bảo mật.

Các kết quả thử nghiệm trên bo mạch DE4 cho thấy, giải pháp trao đổi khóa IKE sử dụng Nios II của nhóm tác giả hoạt động ổn định, đáp ứng các yêu cầu đặt ra. Trong thời gian tới, nhóm sẽ tiếp tục cải tiến tốc độ, cũng như hoàn thiện các chức năng khác để đáp ứng đầy đủ các yêu cầu của bài toán.

TÀI LIỆU THAM KHẢO

[1] Altera Corp, "DE4 User manual", 2016, URL: ftp://ftp.altera.com/up/pub/Altera_Material/Boards/DE4/DE4_User_Manual.pdf (Truy cập ngày 17/8/2021).

[2] Altera Corp, "Triple-Speed Ethernet MegaCore Function User Guide – Altera", 2016, URL: https://www.altera.com/literature/ug/ug_ethernet.pdf Truy cập ngày 17/8/2021).

[3] "IEEE 802.1X-rev-2010, IEEE Standard for Local and metropolitan area networks Port – Base Network Access Control", 2010, URL: https://standards.ieee.org/getieee802/download/802.1X-2010.pdf (Truy cập ngày 17/8/2021).

[4] "RFC 3748: Extensible Authentication Protocol", 2004, URL: https://datatracker-.ietf.org/doc/html/rfc3748 (Truy cập ngày 17/8/2021).

[5] "RFC 4303: IP Encapsulating Security Payload (ESP)", 2005, URL: https://datatracker-.ietf.org/doc/html/rfc4303 (Truy cập ngày 17/8/2021).

[6] "RFC 7296: Internet Key Exchange Protocol Version 2 (IKEv2)", 2014, URL: https://data-tracker.ietf.org/doc/html/rfc7296 (Truy cập ngày 17/8/2021).

[7] Jing Lu, John W. Lockwood, "IPSec implementation on Xilinx Virtex-II pro FPGA and its application", IPDPS 2005.

[8] Zdenek Martinasek, Jan Hajny, M David Smekal, and others, "200 Gbps Hardware Accelerated Encryption System for FPGA Network Cards", ASHES '18: Proceedings of the 2018 Workshop on Attacks and Solutions in Hardware Security, 10/2018.

[9] The OpenSource IPsec-based VPN Solution, URL: https://strongswan.org (Truy cập ngày 17/8/2021).

[10] "RFC 8229: TCP Encapsulation of IKE and IPsec Packets", 2017, URL: https://data-tracker.ietf.org/doc/html/rfc8229 (Truy cập ngày 17/8/2021).

SƠ LƯỢC VỀ TÁC GIẢ

Vũ Tá Cường

Đơn vị công tác: Viện Khoa học – Công nghệ mật mã, Ban Cơ yếu Chính phủ.

Email: vutacuong109@gmail.com

Quá trình đào tạo: Nhận bằng Đại học năm 2011, Thạc sĩ năm 2013 và Tiến sĩ năm 2016 chuyên ngành Vô tuyến kỹ thuật, Đại học Hàng không vũ trụ Kharkov, Ukraina.

Hướng nghiên cứu hiện nay: Kỹ thuật mật mã.

La Hữu Phúc

Đơn vị công tác: Viện Khoa học – Công nghệ mật mã, Ban Cơ yếu Chính phủ.

Email: phucpvkt@hotmail.com

Quá trình đào tạo: Nhận bằng Đại học năm 1998 và Thạc sĩ năm 2002 chuyên ngành Kỹ thuật điện tử, Học viện kỹ thuật mật mã. Nhận bằng Tiến sĩ năm 2015, Viện Khoa học - Công nghệ quân sự.

Hướng nghiên cứu hiện nay: Thiết kế chế tạo máy mã, thiết bị mã chuyên dụng.