Một giải pháp bảo mật cho giao thức Modbus TCP phòng chống tấn công vào hệ thống SCADA sử dụng giao thức này

A security solution for Modbus TCP protocol to prevent attacks on SCADA system using this protocol

Nguyễn Văn Xuân, Nguyễn Tăng Cường, Hoàng Đức Trọng
Học viện Kỹ thuật Quân sự
e-Mail:nguyenvanxuanirf@yahoo.com,ntcuong210@gmail.com,herodtrong@gmail.com

Vũ Đức Tân
Học viện Phòng không – Không quân
e-Mail: vuductan1981@gmail.com

Tóm tắt

Giao thức Modbus được sử dụng rộng rãi trong các hệ thống công nghiệp quan trọng, hệ thống điều khiển giám sát và thu thập dữ liệu (SCADA). Với nhu cầu phát triển, mở rộng ngày càng tăng của mạng công nghiệp, giao thức Modbus cũng được kết nối với Internet sử dụng giao thức TCP/IP hoặc các giao thức truyền tải dữ liệu khác. Sử dụng cơ sở hạ tầng Internet tạo ra những lỗ hổng và nguy cơ dễ bị tấn công gây ra những thiệt hại không mong muốn. Trong bài báo này, chúng tôi trình bày kỹ thuật ứng dụng mật mã khóa đối xứng AES bảo mật cho giao thức Modbus TCP/IP. Giải pháp này làm giảm đáng kể các lỗ hổng, tăng cường tính bảo mật, xác thực, toàn vẹn gói tin cho giao thức Modbus TCP/IP, để phòng chống các cuộc tấn công vào hệ thống SCADA sử dụng giao thức này.

Từ khóa: Bảo mật giao thức Modbus, Modbus TCP/IP, An ninh mạng, Mã hóa AES.

Abstract

Modbus protocol is widely deployed in critical industrial systems or supervisory control system and data acquisition (SCADA). With the increasing demands on industrial networks, the Modbus protocol is also connected over Internet using TCP/IP protocol or other transmission protocol. Using the Internet infrastructure, the systems are likely vulnerable to risks, this cause unpredictabe damage. In this paper, we address a solution of symmetric key cipher AES for high-secured Modbus TCP/IP protocol. This approach is for reducing vulnerabilities, enhancing security, authentication and integrity in Modbus TCP/IP protocol to prevent cyber attacks on SCADA systems using this protocol.

Keywords: Secure Modbus Protocol, Modbus TCP/IP, Cyber security, AES.

Chữ viết tắt

AES	Advanced Encryption Standard
CRC	Cyclic Redundancy Check
DES	Data Encryption Standard
TCP	Transmission Control Protocol
IP	Internet Protocol
LRC	Longitudial Redundancy Check
HMI	Human Machine Interface
SCADA	Supervisory control and data acquisition
PLC	Programmable Logic Controller
PDU	Protocol Data Unit
OSI	Open Systems Interconnection
LAN	Local Area Network
DMZ	Demilitarized Zone
DoS	Denial of Service
$GF(2^n)$	The finite field of order 2^n

1. Đặt vấn đề

Hệ thống điều khiển giám sát và thu thập dữ liệu (SCADA) được sử dụng rộng rãi trong các hệ thống công nghiệp tự động hóa, trong các cơ sở hạ tầng rất quan trọng như các nhà máy hóa chất, trạm phát điện, mạng lưới truyền tải và phân phối điện, mạng lưới phân phối nước và xử lý chất thải, hệ thống thủy lợi, các lò phản ứng tổng hợp hạt nhân… Các hệ thống SCADA như vậy có ý nghĩa chiến lược bất kỳ một lỗi hoặc trục trặc gây ra thì hậu quả thiệt hại là rất lớn. Một hệ thống SCADA cho phép kết nối liên thông giữa nhiều mạng thông qua nhiều loại giao thức, bao gồm cả giao thức Internet (IP) [2]. Phạm vi kết nối rộng này làm cho hệ thống SCADA có thêm nhiều lỗ hổng, cùng với những lỗ hổng sẵn có của hệ thống SCADA làm chúng rất dễ bị tấn công bởi những kẻ có ý đồ xấu.

Trong thiết kế truyền thống các hệ thống SCADA mối quan tâm chính là hiệu suất, hiệu quả của của hệ thống. An ninh trong các hệ thống SCADA ít được quan tâm cho đến khi xuất hiện nhiều sự cố an ninh. Cụ thể tháng 3 năm 2000, các thông tin liên lạc giữa các trạm bơm trong các dịch vụ nước Maroochy ở Úc đã bị mất, các trạm bơm không hoạt động được, các nhân viên phát hiện ra một người nào đó đã tấn công vào hệ thống và gây ra sự cố cho hệ thống này [3]. Vào tháng 8 năm 2003, một con sâu máy tính đã vượt qua tường lửa (firewall) và xâm nhập vào hệ thống điều khiển SCADA tại các nhà máy điện hạt nhân DavisBesse ở Ohio [4]. Xu hướng này lên đến đỉnh điểm với Stuxnet [5], một phần mềm độc hại tấn công vào hệ thống SCADA phát hiện vào tháng 7 năm 2010. Stuxnet là một phần mềm độc hại tấn công tinh

vi vào các loại PLC, sửa đổi kiểm soát chúng gây ra những bất thường trong hệ thống.

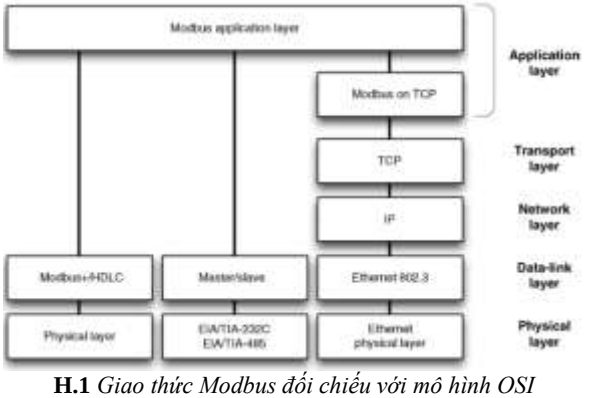
Hệ thống SCADA ban đầu được xây dựng trên giả thiết là tất cả các thành phần hoạt động trong mạng là hợp pháp. Các hệ thống như vậy hầu như không có các biện pháp bảo vệ chống lại các cuộc tấn công có chủ ý. Các thành phần trong mạng không có xác thực danh tính, xác thực truy nhập, cũng không thể xác minh được nội dung của thông điệp có hợp lệ hay không hợp lệ và tất cả các dữ liệu được truyền qua mạng đều là bản rõ, không có bất kỳ mã hóa nào. Giao thức Modbus, Modbus TCP cũng nằm trong số đó, hơn nữa theo xu hướng công nghệ hiện nay thì Modbus TCP đang được sử dụng rộng rãi trong các hệ thống SCADA để giao tiếp giữa các thành phần trong mạng. Từ những lý do đó chúng tôi nghiên cứu ứng dụng tiêu chuẩn mã hóa tiên tiến AES bảo mật cho giao thức Modbus TCP, nhằm ngăn chặn các cuộc tấn công của những kẻ có ý đồ xấu vào hệ thống mạng SCADA sử dụng giao thức này là rất cần thiết.

2. Nội dung chính

2.1 Giao thức Modbus

Modbus [6] được thiết kế vào năm 1979 bởi hãng Modicon (sau này thuộc Schneilder Automation). Hiện nay, Modbus là một trong những giao thức phổ biến nhất được sử dụng trong hệ thống điều khiển công nghiệp. Theo khảo sát hiện tại bởi "American Control Engineering magazine", 40% số hệ thống điều công nghiệp quan trọng được triển khai bằng giao thức Modbus. Thành công của Modbus bắt nguồn từ việc tương đối dễ sử dụng. Nó cũng là một tiêu chuẩn mở, được phân phối miễn phí và được hỗ trợ rộng rãi bởi các thành viên của tổ chức Modbus, vẫn còn hoạt động đến ngày nay.

Theo mô hình OSI thì Modbus thực chất là một chuẩn giao thức và dịch vụ thuộc lớp ứng dụng, có thể được thực hiện vận chuyển qua TCP/IP hay qua đường truyền RS232/RS485 (H.1).

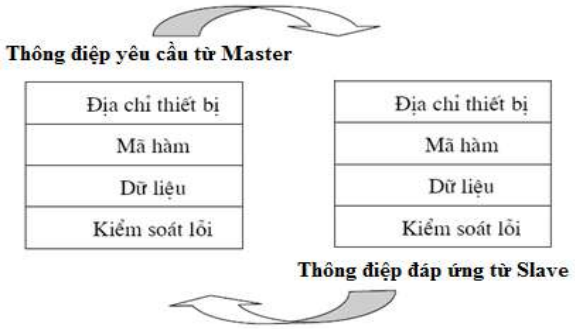


H.1 *Giao thức Modbus đối chiếu với mô hình OSI*

2.1.1 Cơ chế giao tiếp

Giao thức Modbus giao tiếp theo cơ chế yêu cầu/đáp ứng, trong đó chỉ có một thiết bị chủ (Master) có thể gửi yêu cầu, còn các thiết bị tớ (Slave) đáp ứng bằng dữ liệu trả lời hoặc thực hiện một hành động nhất định theo yêu cầu (H.2). Kết hợp mã hàm (function code) và dữ liệu, Modbus thực hiện được một loạt các lệnh như:

- Đọc giá trị một thanh ghi
- Ghi giá trị cho một thanh ghi
- Đọc giá trị một nhóm các thanh ghi liên tiếp
- Ghi giá trị cho một nhóm thanh ghi liên tiếp
- Đọc các trị đầu vào
- Ghi các giá trị đầu ra



H.2 *Cơ chế giao tiếp của giao thức Modbus*

Địa chỉ của các trạm trong mạng Modbus từ 0-247, trong đó địa chỉ 0 là địa chỉ gửi quảng bá.

2.1.2 Cấu trúc một thông điệp Modbus

Modbus có hai chế độ truyền là RTU và ASCII. Trong chế độ truyền RTU (Modbus RTU, H.3) mỗi byte trong thông điệp được truyền đi là 8 bit nhị phân, khởi đầu và kết thúc bức điện là khoảng nghỉ ít nhất 4 chu kỳ ký tự. Chế độ truyền ASCII (Modbus ASCII, H.4) mỗi byte trong thông điệp được truyền đi bởi hai ký tự ASCII là hai ký tự số số hexa biểu diễn byte đó, ký tự khởi đầu là ':', hai ký tự kết thúc là CR và LF.

Mỗi thông điệp đều được định dạng đơn giản gồm phần khởi đầu, kết thúc, địa chỉ, mã hàm, dữ liệu, mã kiểm lỗi để kiểm tra thông điệp có bị lỗi truyền không.

Khởi đầu	Địa chỉ	Mã hàm	Dữ liệu	Mã CRC	Kết thúc
(----)	8 bit	8 bit	nx8 bit	16 bit	(----)

H.3 *Cấu trúc bức điện Modbus RTU*

Khởi đầu	Địa chỉ	Mã hàm	Dữ liệu	Mã LRC	Kết thúc
1ký tự :	2 Ký tự	2 Ký tự	n Ký tự	2 Ký tự	2 Ký tự CR+LF

H.4 *Cấu trúc bức điện Modbus ASCII*

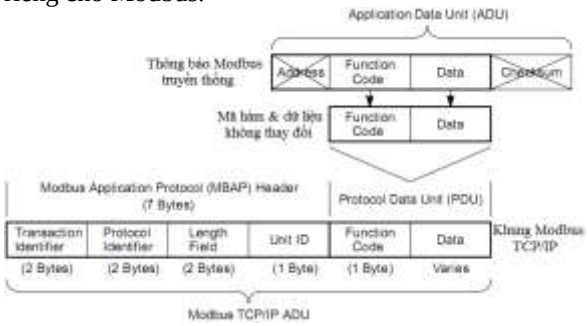
2.1.3 Modbus TCP/IP

Modbus TCP/IP (Modbus TCP) [7] chỉ đơn giản là giao thức Modbus RTU sử dụng giao diện TCP để truyền tải trên nền Ethernet. Tức họ giao thức TCP/IP được dùng để truyền chính xác các thông điệp của giao thức Modbus. Modbus TCP nhúng một thông điệp (khung dữ liệu) Modbus chuẩn vào phần dữ liệu của gói TCP mà không có sự kiểm tra, xác thực giao thức Modbus, xem H.5.

Phần kiểm lỗi (checksum) của Modbus không được sử dụng thay vào đó sử dụng kiểm lỗi của TCP/IP để bảo toàn dữ liệu. Mã hàm và dữ liệu của Modbus được chuyển thành phần dữ liệu của TCP, không có bất kỳ thay đổi nào và bổ sung thêm 7 byte ở phía trước chia thành bốn trường như sau:

- **Transaction Identifier** (2bytes): Được sử dụng để phân biệt các thông điệp khi có nhiều thông điệp khác nhau truyền đi từ một kết nối TCP.
- **Protocol Identifier** (2bytes): Trường này luôn đặt bằng 0 với Modbus, các giá trị khác mở rộng cho tương lai.
- **Length** (2 bytes): Cho biết độ dài của các trường còn lại gồm Unit ID, Function code, Data.
- **Unit ID** (1byte): Có giá trị từ 0-255 sử dụng để nhận dạng các thiết bị trong mạng Modbus TCP có ý nghĩa như trường địa chỉ (address) trong Modbus chuẩn.

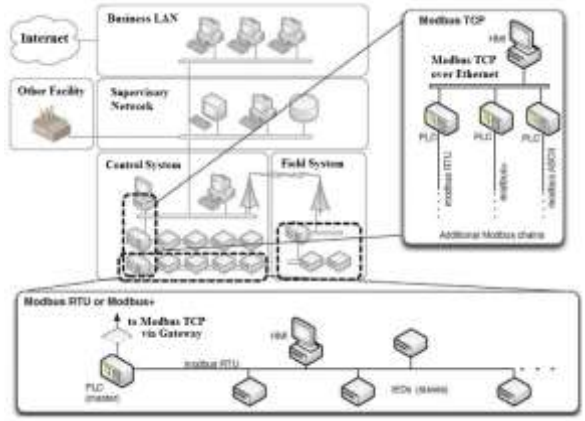
Dữ liệu của Modbus TCP sau khi được tạo ra như H.5 được giao thức TCP gửi và nhận qua cổng 502 dành riêng cho Modbus.



H.5 *Tạo một thông điệp Modbus TCP*

2.1.4 Ứng dụng Modbus trong mạng công nghiệp

- Modbus thường được triển khai giao tiếp giữa PLC và HMI, hoặc giữa PLC chủ và các thiết bị tớ như PLCs, HMIs, Drivers, Sensors, I/O devices, … hỗ trợ tối đa 247 thiết bị trong một bus xem H.6.
- Một triển khai phổ biến khác sử dụng Modbus TCP/IP là trong vùng DMZ SCADA (vùng cho truy cập từ bên ngoài), hoặc trong mạng LAN giám sát có HMI chủ có khả năng quản lý tập trung một số PLC chủ, mỗi PLC chủ trong số đó có thể kết nối trên bus của nó các thiết bị khác, để thực hiện một vòng quét xem H.6.



H.6 *Modbus sử dụng trong mạng công nghiệp*

2.1.5 Những vấn đề lo ngại về an ninh

- Thiếu xác thực: Modbus, Modbus TCP chỉ yêu cầu sử dụng đúng địa chỉ, mã hàm và dữ liệu liên quan, chúng không thể xác minh được một thông điệp có hợp lệ hay không hợp lệ nên rất dễ bị tấn công giả mạo thông điệp, sửa đổi thông điệp...
- Các thông điệp không được mã hóa: Địa chỉ và mã hàm trong thông điệp được truyền đi ở dạng rõ ràng do đó có thể dễ dàng giả mạo, bắt giữ, sửa đổi hoặc gửi lặp lại thông điệp nhiều lần. Có thể chặn bắt thông tin truyền thông đến/đi từ một thiết bị Modbus bất kỳ trong mạng cũng có thể tiết lộ những thông tin quan trọng liên quan đến cấu hình và sử dụng thiết bị.
- Modbus TCP không có kiểm lỗi (checksum) cho mỗi thông điệp ở tầng ứng dụng nên tạo một lệnh giả mạo càng dễ dàng hơn chỉ cần có mã hàm và dữ liệu, checksum sẽ được tự động tạo ra ở tầng vận chuyển TCP.
- Không có cơ chế ngăn chặn gửi thông điệp quảng bá, tất cả các thiết bị trong mạng Modbus đều nhận được một thông điệp phát đi có nghĩa là một thông điệp quảng bá có thể được sử dụng cho tấn công từ chối dịch vụ DoS.

Những mối lo ngại an ninh trình bày trong mục này sẽ được khắc phục khi bổ sung thêm bảo mật, chứng thực cho giao thức, mục 2.3 sẽ giải quyết vấn đề này.

2.2 Tiêu chuẩn mã hóa tiên tiến AES

Từ thời cổ đại cho đến ngày nay mật mã luôn là nền tảng cơ bản của an toàn thông tin [8]. Mật mã AES (Advanced Encryption Standard, hay Tiêu chuẩn mã hóa tiên tiến) là một thuật toán mã hóa khối được chính phủ Hoa kỳ áp dụng làm tiêu chuẩn mã hóa. Giống như tiêu chuẩn tiền nhiệm DES, AES được áp dụng rộng rãi trên toàn thế giới. AES và DES đều là thuật toán mã hóa khóa đối xứng (mã hóa, giải mã cùng một khóa), DES với khóa 56 bít, AES với khóa 128 bít, tuy nhiên DES và biến thể của nó như 3DES (mã hóa DES ba lần) hiện nay có độ an toàn không cao do khóa ngắn dễ bị tấn công vét cạn nên AES đang dần thay thế cho DES và 3DES.

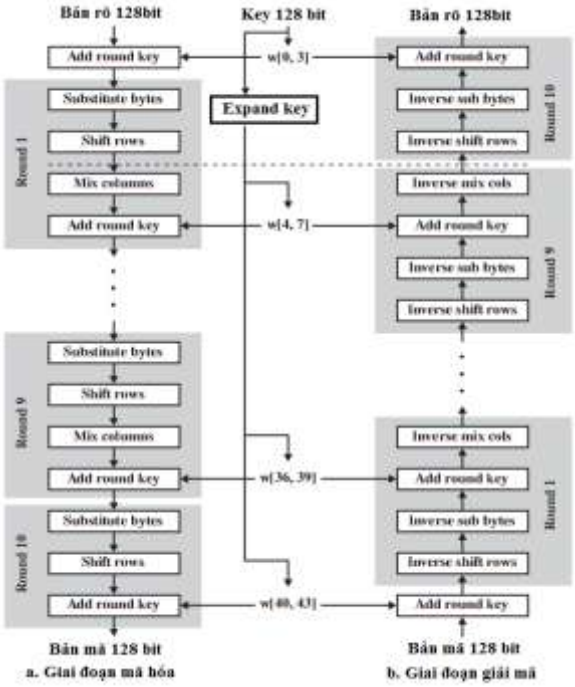
Nền tảng cơ bản của các thuật toán mã hóa khóa đối xứng hiện đại nói chung, mã hóa DES, AES nói riêng sử dụng phép biến đổi thay thế S-Box và phép hoán vị P, hai phép toán này được lặp đi lặp lại nhiều lần. DES lặp lại trong 16 vòng, AES 128, 196, 256 bít lần lượt lặp lại trong 10, 12, 14 vòng. Kết hợp hai phép biến đổi thay thế S-Box và phép hoán vị P tạo ra hai tính chất quan trọng của mã hóa là tính khuếch tán (diffusion) và tính gây lẫn (confusion).

- Tính khuếch tán: Một bít của bản rõ tác động đến tất cả các bít của bản mã, hay nói cách khác một bít của bản mã chịu tác động của tất cả các bít trong bản rõ. Tính chất này làm giảm tối đa mối liên quan giữa bản rõ và bản mã, ngăn chặn việc suy ra khóa.
- Tính gây lẫn: Làm phức tạp hóa mối liên quan giữa bản rõ, bản mã và khóa. Do đó cũng ngăn chặn việc suy ra khóa khi có bản mã.

Mã hóa AES (chi tiết xem [9]) sử dụng 4 phép biến đổi chính để mã hóa một khối dữ liệu 128 bit là: Add row key, Substitute bytes, Shift rows, Mix columns. Mỗi phép biến đổi đều nhận tham số đầu vào có kích thước 128 bít và cho ra kết quả cũng có kích thước 128 bít. AES thực hiện 4 phép biến đổi trên trong 10 vòng lặp như bên trái H.7 là giai đoạn mã hóa, bên phải thực hiện ngược lại là giai đoạn giải mã.

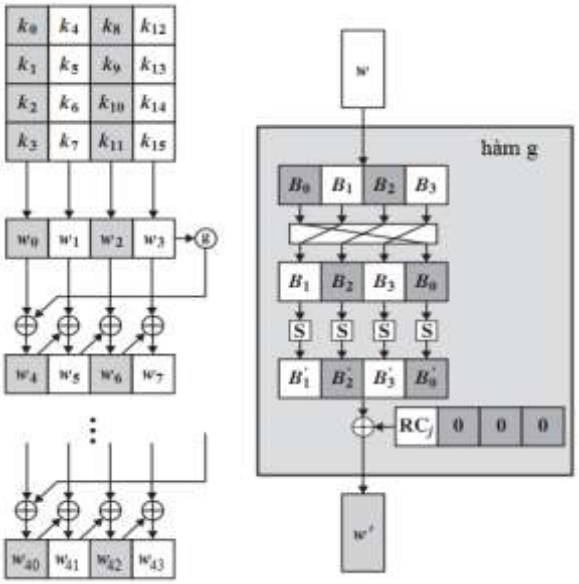
Các phép biến đổi Substitute bytes, Shift rows, Mix columns có phép biến đổi ngược tương ứng là Inverse sub bytes, Inverse shift rows, Inverse mix cols. Riêng phép biến đổi Add row key đơn giản chỉ là phép XOR nên phép biến đổi ngược cũng là Add row key. Các phép biến đổi ngược bên phải H.7 giải mã AES cũng gồm 10 vòng thực hiện theo chiều ngược lại.

Kích thước khóa ban đầu là 128 bít (16 byte). AES dùng hàm mở rộng khóa (Expand key) để mở rộng kích thước khóa thành 44 word 32 bít. 44 word này được chia thành 11 cụm khóa con, mỗi khóa con (128 bít) hay 4 word làm tham số vào cho 11 thao tác Add row key. 128 bit bản rõ và 128 bit khóa con đều được tổ chức thành khối ma trận 4x4 mỗi phần tử là 1 byte được sắp xếp lần lượt theo cột.



H.7 *Quá trình mã hóa/giải mã AES.*

- Phép biến đổi Add row key: Khóa con được kết hợp với các khối dữ liệu vào, mỗi khóa con có độ dài giống như dữ liệu vào. Quá trình kết hợp được thực hiện bằng cách XOR từng bít của khóa con với từng bít khối dữ liệu vào.
- Substitute bytes: Các byte của khối đầu vào được thay thế bằng tra trong bảng S-Box có kích thước (16x16) byte. Đây chính là quá trình phi tuyến của thuật toán. Hộp S-box được tạo ra từ một phép biến đổi khả nghịch trong trường hữu hạn GF (2^8) có tính chất phi tuyến. Tương tự phép biến đổi nghịch Inverse sub bytes thực hiện tra trong bảng IS-BOX. Chi tiết các bảng S-BOX, IS-BOX, trường hữu hạn GF(2^8) xem tài liệu [9].
- Shift rows: Các hàng của khối dữ liệu vào được dịch vòng một số bước nhất định. Hàng đầu được giữ nguyên, mỗi byte của hàng thứ 2 được dịch vòng trái một vị trí. Tương tự các hàng thứ 3 và 4 được dịch vòng 2 và 3 vị trí. Do vậy, mỗi cột của khối đầu ra của bước này sẽ bao gồm các byte ở đủ 4 cột khối đầu vào. Phép biến đổi ngược Inverse shift rows thực hiện ngược lại, dòng 1 giữ nguyên, các dòng 2, 3 và 4 được dịch vòng phải tương ứng 1 byte, 2 byte và 3 byte.
- Mix columns: Mỗi cột của khối dữ liệu vào (ma trận 4x4) tương ứng là hệ số của một đa thức f(x) bậc bằng 3 (với hệ số tự do ứng byte đầu tiên của cột). Đa thức này được nhân với đa thức $a(x)=3x^3+x^2+x+2$, sau đó thực hiện phép chia modulo cho đa thức $n(x)=x^4+1$, kết quả phép modulo là đa thức bậc 3 $c(x)=f(x).a(x) \bmod n(x)$, bốn byte hệ số của đa thức c(x) được thay thế tương ứng cho 4 byte trong cột tạo đa thức f(x) của khối dữ liệu. Các phép cộng và nhân trong đa thức được thực hiện trong trường GF(2^8). Phép biến đổi ngược Inverse mix cols, thực hiện tương tự mỗi cột của khối dữ liệu vào được nhân với đa thức $b(x) = 11x^3+13x^2+9x+14$, sau đó cũng thực hiện phép chia modulo cho đa thức $n(x)=x^4+1$, cuối cùng thu được bốn hệ số của đa thức kết quả, các hệ số này thay thế cho cột tương ứng. Chi tiết chứng minh phép biến đổi Inverse mix cols là phép biến đổi ngược của Mix columns xem tài liệu [9].
- Hàm mở rộng khóa (Expand key). 16 byte khóa ký hiệu là: k0, k1,k2…k15 và được sắp xếp lại thành ma trận 4x4 sau đó phép mở rộng khóa (tạo khóa con) được thực như trong H.8.



H.8 *Thuật toán tạo khóa con w0 đến w44*

Từ bốn từ (word) đầu vào w0w1w2w3, w0 = (k0,k1,k2,k3)…w3=(k12,k13,k14,k15) trong lần lặp đầu tiên thao tác sinh ra bốn word w4w5w6w7, lần lặp thứ 2, từ w4w5w6w7 sinh ra w8w9w10w11, cứ như thế cho đến lần lặp thứ 10 sinh ra bốn word cuối cùng w40w41w42w43 như H.8.

Trong mỗi lần lặp để sinh ra 4 word, thì word đầu tiên sinh ra theo quy tắc $w_i = w_{i-4} \oplus g$, với g =SubWord(RotWord(w_{i-4},w_{i-3},w_{i-2},w_{i-1})) $\oplus$ Rcon[i/4], i=4,8,12..40. Ba word tiếp theo sinh ra theo quy tắc $w_j = w_{j-1} \oplus w_{j-4}$, j=i+1,i+2,i+3.

+ RotWord: Dịch vòng trái một byte. Giả sử word đầu vào có 4 byte là [b0, b1, b2, b3] thì kết quả của RotWord là [b1, b2, b3, b0].

+ SubWord: Thay thế mỗi byte trong word đầu vào bằng cách tra cứu bảng S-Box trong thao tác Substitute Bytes.

+ Rcon: Là một mảng hằng số. Mảng này gồm 10 word ứng với 10 vòng AES. Bốn byte của một phần tử Rcon[j] là (RC[j], 0, 0, 0) với RC[j] là mảng 10 byte hexa như sau:

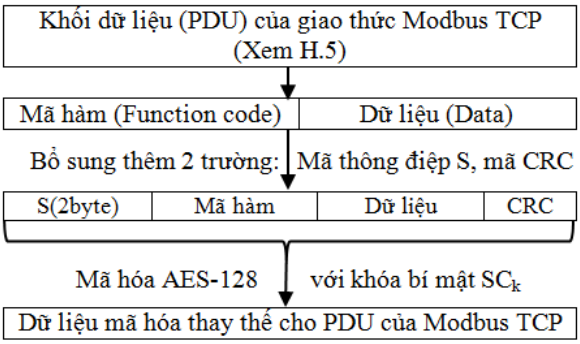
j	1	2	3	4	5	6	7	8	9	10
RC[j]	01	02	04	08	10	20	40	80	1B	36

2.3 Bảo mật cho giao thức Modbus TCP

Carcano[1] đã chứng minh thiếu xác thực trong giao thức SCADA tạo cơ hội dễ dàng cho các mã độc hại tấn công vào các cảm biến, cơ cấu chấp hành của hệ thống SCADA. Tường lửa và các hệ thống phát hiện xâm nhập có thể bảo vệ, phát hiện những tấn công vào hệ thống SCADA. Tuy nhiên, các Hacker cũng luôn cố gắng tạo ra các mã độc hại có thể vượt qua những vòng kiểm soát an ninh này. Do vậy cách tốt nhất để giải quyết các mối đe dọa an ninh đó là giải quyết tận gốc bằng cách thiết kế lại giao thức có bảo mật và độ an toàn cao. Nhưng một giải pháp như vậy là rất khó thực hiện bởi vì nó đòi hỏi những thay đổi đáng kể đối với các kiến trúc và cấu hình hệ thống điều khiển. Thay vào đó, chúng tôi đề xuất một cách tiếp cận thực tế hơn, bổ sung thêm cơ chế bảo mật cho một giao thức để khắc phục các lỗ hổng. Trong bài báo này chúng tôi lựa chọn giao thức Modbus TCP được sử dụng phổ biến trong hệ thống SCADA để nghiên cứu. Mục đích bảo mật cho giao thức Modbus TCP là nhằm đáp ứng các yêu cầu bảo mật sau đây:

- Bí mật: Chỉ có người nhận đã xác thực có thể lấy ra được nội dung của thông tin chứa đựng trong thông điệp.
- Xác thực: Người nhận cần có khả năng xác định người gửi và kiểm tra xem người gửi đó có thực sự gửi thông tin đi hay không.
- Toàn vẹn: Người nhận cần có khả năng xác định thông tin có bị thay đổi bởi bên thứ ba nào hay không.
- Chống lặp lại: Không cho phép bên thứ ba nào sao chép lại thông điệp và gửi nhiều lần đến người nhận mà người nhận không biết.
- Không từ chối: Người gửi không thể từ chối việc mình đã gửi thông tin đi.

Cơ chế bảo mật trên giao thức Modbus TCP được thực hiện như sau (H.9):



H.9 *Quá trình bảo mật cho giao thức Modbus TCP*

- Trường S (độ dài 2byte) mã số định danh thông điệp được Master chèn vào, mỗi thông điệp gửi đi, có một mã số khác nhau, mã số này sẽ giúp chống lại các tấn công phát lại thông điệp.
- Trường CRC: Là hàm băm CRC 16 bit (với đa thức sinh là: $x^{16} + x^2 + x$) của các trường: S, mã hàm, dữ liệu. 16 bit mã CRC được chèn vào cuối thông điệp để tạo cơ chế xác thực và chống sửa đổi gói tin. Lưu ý các thông điệp của Modbus đã được họ giao thức TCP/IP truyền tải đảm bảo chính xác không bị lỗi nên mục đích của trường CRC ở đây không để kiểm tra lỗi truyền thông điệp mà dùng cho mục đích xác thực, phát hiện sửa đổi thông điệp.

Giao thức Modbus TCP thực hiện trao đổi thông điệp giữa Master và Slave theo cơ chế yêu/đáp ứng như H.2, Master thực hiện mã hóa toàn bộ các trường S, mã hàm, dữ liệu và mã CRC bằng khóa bí mật SC_k (chỉ Master và Slave có), rồi gửi cho Slave. Slave nhận được thông điệp mã hóa tiến hành giải mã bằng khóa SC_k, sau đó tính lại CRC so sánh với CRC của thông điệp nhận được, tiếp đến kiểm tra mã số định danh thông điệp S có trùng với thông điệp cũ không, tất cả đều hợp lệ thì lọc ra mã hàm, dữ liệu để quyết định trả lời yêu cầu của Master. Slave trả lời Master cũng tiến hành tương tự như Master gửi yêu cầu cho Slave chỉ khác mã định danh thông điệp không được Slave tạo ra mà lấy từ thông điệp của Master và tăng thêm một đơn vị. Quá trình giao tiếp giữa Master và Slave thực hiện như H.9 sẽ đảm bảo được một số tính chất sau:

- **Tính bí mật**: Tất cả thông điệp trao đổi giữa Master và Slave đều được mã hóa AES-128bit chỉ có Master, Slave có khóa bí mật SC_k nên bất kỳ bên thứ ba nào có được thông điệp trung gian đều không thể giải mã được nội dung do không có khóa SC_k.
- **Tính chứng thực**: Được thực hiện do kết hợp giữa mã CRC của thông điệp và mã hóa toàn bộ thông điệp trước khi truyền đi. Khi Slave nhận được thông điệp tiến hành giải mã và tính lại CRC của thông điệp nếu khớp với CRC của thông điệp gửi đi thì chắc chắn thông điệp đó là của Master. Tính chứng thực này chống lại được tất cả các hình thức tấn công mạo danh Master gửi lệnh cho Slave, vì giả sử một bên thứ ba nào đó mạo danh Master gửi thông điệp cho Slave do không có khóa bí mật SC_k nên bên thứ ba phải gửi một thông điệp theo truyền thống hoặc theo sơ đồ H.9 với khóa bí mật nào đó khác SC_k hoặc gửi thông điệp bất kỳ nào đó khác, khi Slave nhận được thông điệp đó tiến hành giải mã với khóa bí mật SC_k, tính lại CRC so sánh với CRC của thông điệp sẽ thấy không khớp nên biết được thông điệp không phải do Master gửi và loại bỏ. Xác suất bên thứ ba gửi ngẫu nhiên một thông điệp để Slave giải mã ra rồi tính CRC khớp với CRC của thông điệp là rất rất nhỏ.
- **Tính toàn vẹn gói tin:** Giả sử một bên thứ ba nhận được thông điệp trung gian (bản mã) muốn sửa đổi nội dung thông điệp nhưng do không có khóa bí mật SC_k để giải mã, rồi sửa nội dung, tính lại CRC, mã hóa lại gửi cho Slave nên buộc phải sửa đổi bản mã theo một cách nào đó khác do vậy khi Slave nhận

được thông điệp này. Slave giải mã và tính lại CRC sẽ thấy không khớp với CRC của thông điệp gửi đi nên biết được nội dung thông điệp đã bị thay đổi.

- **Chống lặp lại**: Mỗi một thông điệp gửi đi, trước khi mã hóa Master chèn mã số định danh thông điệp (khác nhau mỗi lần truyền) vào trường S. Do đó nếu một bên thứ ba nào đó nhận được một thông điệp trung gian, giả mạo Master phát lại thông điệp đó cho Slave, Slave nhận được hai thông điệp có cùng số định danh thì loại bỏ thông điệp thứ hai. Cũng do chèn thêm mã số định danh thông điệp khác nhau cho mỗi lần truyền, mà hai thông điệp Master gửi đi giống nhau nhưng chỉ khác nhau mã số này, sẽ cho bản mã hoàn toàn khác nhau (tính chất phân tán của AES) nên gây khó khăn cho bên thứ ba muốn giải mã, thu thập thông tin từ các thông điệp.
- **Không từ chối**: Do cả Master và Slave đều có khóa bí mật SC_k nên về lý thuyết một thông điệp gửi đi không kết luận được chính xác của Master hay Slave. Tuy nhiên trong hệ thống SCADA sử dụng Modbus TCP giao tiếp theo cơ chế yêu cầu/đáp ứng nên các Slave luôn bị động không gửi bất cứ thông điệp nào nếu Master không yêu cầu nên một Slave nhận được thông điệp yêu cầu có địa chỉ, mã định danh thông điệp S, CRC… hợp lệ chứng tỏ thông điệp đó phải do Master gửi đi (Master không thể từ chối đã gửi yêu cầu này).

2.4 Cài đặt và thử nghiệm giao thức bảo mật Modbus TCP.

Mô hình hệ thống thử nghiệm giao thức bảo mật Modbus TCP/IP cho hệ SCADA như sau:

H.10 *Mô hình thử nghiệm giao thức bảo mật Modbus TCP*

Trong mô hình H.10 máy tính đóng vai trò là thiết bị chủ (Master). Board nhúng ARM Tiny 6410 là thiết bị tớ (Slave) giao tiếp với máy tính qua giao thức bảo mật Modbus TCP (xem mục 2.3) trên nền mạng LAN. Board nhúng ARM kết nối với biến tần VFD-B của Delta qua chuẩn RS458 và giao thức Modbus RTU. Biến tần VFD-B của Delta được cấu hình để sử dụng truyền thông Modbus RTU chuẩn điều khiển động cơ ba pha, mô hình thức tế xem H.11.

H.11 *Mô hình thực tế thử nghiệm bảo mật Modbus TCP*

2.4.1 Chương trình trên máy tính

Chương trình giám sát và điều khiển trên máy tính được thiết kế trên C# 2008 của bộ phần mền Microsoft Visual Studio, thực hiện điều khiển, giám sát một số chức năng của biến tần VFD-B như sau:

+ Gửi lệnh bật/tắt động cơ tới biến tần
+ Gửi lệnh đặt tần số ra cho biến tần
+ Gửi lệnh đọc dòng, áp, tần số ra của biến tần

Các lệnh trên đều được gửi đi theo giao thức bảo mật Modbus TCP. Để mã hóa phần dữ liêu của Modbus TCP, sử dụng các hàm đã được chuẩn hóa cho mã hóa AES trong thư viện "System.Security.Cryptography" của C# [10], để gửi/nhận gói tin Modbus TCP sử dụng các hàm trong viện nModbus [11].

Từ tài liệu của biến tần VFD-B [12] có một số lệnh điều khiển và giám sát sau:

Chức năng	Mã hàm	Địa chỉ thanh ghi	Giá trị
Bật động cơ	06H	2000H	0002H
Tắt động cơ	06H	2000H	0001H
Đặt tần số	06H	2001H	Giá trị cần đặt(0->60)

Bảng 1 *Một số thao tác điều khiển*

Chức năng	Mã hàm	Địa chỉ thanh ghi	Số từ cần đọc
Đọc tần số ra	03H	2103H	0001H
Đọc dòng ra	03H	2104H	0001H
Đọc điện áp ra	03H	2106H	0001H

Bảng 2 *Một số thao tác đọc*

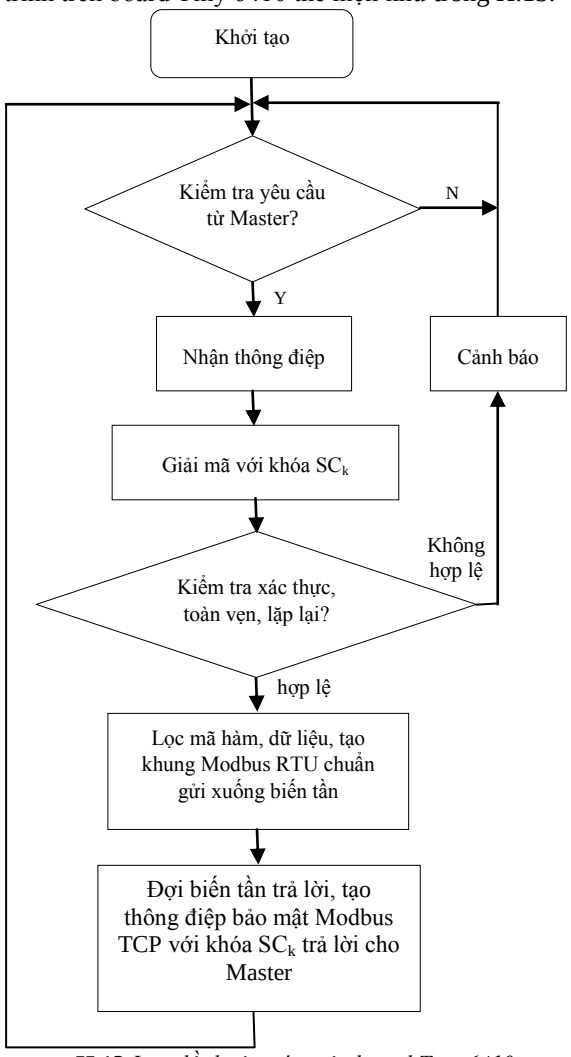
Trong hai bảng 1,2 phần mã hàm, địa chỉ thanh ghi, giá trị/số từ cần đọc, chính là phần dữ liệu (PDU) của Modbus TCP. Như vậy độ dài của phần PDU chỉ có 5 byte nhưng một khối dữ liệu mã hóa bằng AES tối thiểu 128 bit (16 byte) do vậy kết hợp H.9 cần thực hiện như sau: Trước tiên chèn mã định danh thông điệp 2 byte, chèn tiếp 5 byte dữ liệu với thao tác tương ứng như trong bảng 1,2 ở trên, tiếp đó chèn thêm 7 byte giá trị bằng không, tính CRC 16 bít của 14 byte này rồi chèn vào cuối ta được khối dữ liệu 16 byte ký hiệu là M. Khối dữ liệu M được mã hóa AES với khóa bí mật SC_k cho ra bản mã C. Bản mã C tiếp tục được bổ sung thêm 7 byte như H.5 cuối cùng thông điệp này được gửi cho Slave (Board Tiny 6410) qua giao thức TCP/IP với cổng 502.

Giao diện chương trình điều khiển, giám sát trên máy tính được thiết kế như H.12 dưới đây:

H.12 *Giao diện điều khiển trên máy tính*

2.4.2 Chương trình trên Board Tiny 6410

Tiny6410 [14] là một board nhúng được sản xuất và phân phối bởi hãng FriendlyARM. Board này có bộ vi xử lý Samsung S3C6410A ARM1176JZF-S, xung nhịp tối đa đạt 667 MHz, với RAM: 256 MB DDR RAM, và 2GB bộ nhớ NAND Flash. Board nhúng được cài hệ điều hành mã nguồn mở Linux 2.6 và một số thư viện hỗ trợ để chạy được các ứng dụng viết trên QT4.7. Chương trình giao tiếp giữa board Tiny 6410 với máy tính qua mạng LAN sử dụng giao thức TCP/IP và chương trình mã hóa/giải mã AES đều được lập trình trên QT bản 4.7 cài đặt trên hệ điều hành Ubuntu 12.4. Lưu đồ thuật toán của chương trình trên board Tiny 6410 thể hiện như trong H.13.



H.13 *Lưu đồ thuật toán trên board Tiny 6410*

Chức năng chính của board Tiny6410 thực hiện kết nối với máy tính qua mạng LAN, để nhận toàn bộ lệnh từ máy tính (Master) gửi xuống bằng giao thức bảo mật Modbus TCP/IP. Sau đó tiến hành giải mã, kiểm tra tính xác thực, toàn vẹn và kiểm tra thông điệp có bị lặp lại không (xem mục 2.3). Nếu một trong các yêu cầu kiểm tra đó không hợp lệ thì hiển thị cảnh báo lên màn hình LCD và loại bỏ thông điệp đó. Nếu tất cả các kiểm tra là hợp lệ tiến hành lọc lấy mã hàm và dữ liệu trong thông điệp tạo gói Modbus RTU chuẩn sau đó gửi thông điệp chuẩn này cho biến tần, đợi biến tần trả lời. Sau khi có dữ liệu trả lời từ biến tần, lấy thông tin đó tạo khung bảo mật Modbus TCP (như H.9) trả lời cho Master - máy tính.

2.4.3 Một số kết quả thử nghiệm

Đặt chu kỳ quét của chương trình điều khiển giám sát trên máy tính là 500ms, thời gian time out: 1200ms. Kết quả chương trình giám sát thu được và hiển thị trên giao diện: Tần số, điện áp, dòng điện đọc từ biến tần khớp với thông tin xem trên biến tần không xảy ra lỗi nào. Trong quá trình đó trên giao diện có thể liên tục thay đổi tốc độ động cơ hoặc bật tắt nó bằng cách di chuyển thanh trượt để gửi lệnh đặt lại tần số ra của biến tần, hoặc nhấn vào nút start, stop.

So sánh thời gian đáp ứng, độ dài các thông điệp của hai giao thức Modbus TCP và giao thức Modbus TCP có bảo mật như trong bảng 3,4.

Các tham số	Modbus TCP	Bảo mật Modbus TCP
Chu kỳ quét	500ms	500ms
Thời gian time out	1200ms	1200ms
Thời gian đáp ứng	29ms	32ms

Bảng 3 *Thời gian đáp ứng thông điệp*

Thời gian đáp ứng thông điệp tính từ lúc Master bắt đầu gửi thông điệp yêu cầu cho đến khi nhận xong thông điệp trả lời.

Các hàm chức năng	Modbus TCP	Bảo mật Modbus TCP
Mã hàm 06 với gói yêu cầu	12 byte	23 byte
Mã hàm 06 với gói trả lời	12 byte	23 byte
Mã hàm 03 với gói yêu cầu	12 byte	23 byte
Mã hàm 03 với gói trả lời	11 byte	23 byte

Bảng 4 *Độ dài thông điệp gửi cho gói TCP*

Bảng 3 cho thấy độ trễ đáp ứng 29 ms cho Modbus TCP, 32ms bảo mật Modbus TCP. Một sự khác biệt không quá lớn (3ms) cho chu kỳ quét 500 ms và thời gian timeout 1200 ms, độ trễ này sẽ giảm đi nếu thiết bị thử nghiệm có tốc độ cao hơn board Tiny 6410.

Bảng 4 So sánh kích thước phần dữ liệu của Modbus TCP và gói bảo mật Modbus TCP cho hai mã hàm 03, 06. Các gói Modbus TCP có bảo mật lớn hơn so với các gói tin Modbus TCP thường tương ứng. Tuy nhiên, kích thước tăng lên này không phải là một vấn đề trọng yếu ngay cả đối với các mạng SCADA với băng thông thấp.

Thử nghiệm tấn công giả mạo khi không có khóa bí mật SC_k đơn giản như sau. Bằng cách thay đổi khóa bí mật trên giao diện rồi nhấn các nút start, stop quan sát không thấy động cơ thay đổi trạng thái và trên màn hình LCD của board Tiny 6410 hiển thị thông báo nhận được gói tin không hợp lệ điều này chúng tỏ tấn công giả mạo Master đã được phát hiện.

Tiếp tục sử dụng tính năng debug của C#, để sửa đổi nội dung bảm mã, thay bản mã bằng thông điệp rõ, sau đó mới gửi yêu cho board Tiny 6410, ghi lại toàn bộ bản mã rồi gửi lại cho board Tiny 6410 đều cho kết

quả tương tự như trường hợp tấn công giả mạo đơn giản ở trên.

3. Kết luận

Kết hợp tiêu chuẩn mã hóa tiên tiến AES và các yêu cầu của một hệ thống truyền thông an toàn, các tác giả đã xây dựng và thử nghiệm thành công giao thức Modbus TCP có thêm bảo mật, xác thực ở tầng ứng dụng có khả năng khắc phục các lỗ hổng bảo mật của giao thức. Mỗi thông điệp của giao thức đều được mã hóa, xác thực chống lại các dạng tấn công như xem trộm, giả mạo thiết bị chủ gửi thông điệp cho thiết bị tớ, sửa nội dung thông điệp trước khi thiết bị tớ nhận được, gửi lặp lại thông điệp cho thiết bị tớ nhiều lần. Giao thức Modbus TCP có bảo mật ở tầng ứng dụng, được thử nghiệm chạy tốt trên hệ thống mạng gồm một máy tính (Master) kết nối mạng LAN với board Tiny 6410 (Slave), board Tiny được kết nối với biến tần VFD-B của Delta, biến tần được sử dụng để điều khiển động cơ ba pha.

Tài liệu tham khảo

[1] A. Carcano, I. Nai Fovino, M.Masera and A. Trombetta. *SCADA malware: A proof of concept.* Presented at the Third International Workshop on Critical Information Infrastructure Security, 2008.

[2] V. M. Igure and R. D. Williams. *Security and SCADA protocols*. 5th International Topical Meeting on Nuclear Plant Instrumentation Controls, and Human Machine Interface Technology (NPIC and HMIT), pp. 560–567, USA, 2006.

[3] J. Slay and M. Miller. *Lessons learned from the Maroochy Water Breach*. Critical Infrastructure Protection, vol. 253, pp. 73–82, 2008.

[4] D. Ryu, H. Kim and K. Um. *Reducing security vulnerabilities for critical infrastructure*. Journal of Loss Prevention in the Process Industries, vol. 22, pp. 1020–1024, 2009.

[5] N. Falliere, L. O. Murchu and E. Chien, "*W32.Stuxnet Dossier*," Symantec Report version 1.3, Nov 2010.

[6] Modbus IDA. *Modbus application protocol specification v1.1a*, June 4, 2004.

[7] Modbus IDA. *Modbus messaging on TCP/IP implementation guide v1.0a*, June 4, 2004.

[8] Simon Singh. *The Code Book.* Published in New York in 1999.

[9] William Stallings. *Cryptography and Network Security Principles and Practices*, 4th-Edition, Prentice Hall, 2005.

[10] https://msdn.microsoft.com/en-us/library/system.security.cryptography(v=vs.110).aspx

[11] http://ftp.icpdas.com/pub/cd/8000cd/napdos/modbus/nmodbus/nmodbus_api_manual_v1.2_en.pdf

[12] http://prom-electric.ru/data/uploads/manuals/delta_vfd-b_manual_en.pdf

[13] Hoàng Minh Sơn. *Mạng truyền thông công nghiệp*, Nhà bản khoa học kỹ thuật, 2004.

[14] http://www.friendlyarm.net/products/tiny6410

Nguyễn Văn Xuân sinh năm 1981, nhận bằng kỹ sư Công nghệ thông tin năm 2005, nhận bằng Thạc sỹ chuyên ngành Tự động hóa năm 2010 tại Học viện KTQS. Hiện là giảng viên tại Học viện KTQS, là NCS chuyên ngành Tự động hóa, Các hướng nghiên cứu chính là: Thiết kế và lập trình các hệ thống nhúng, Mạng truyền thông công nghiệp và an ninh cho mạng công nghiệp.

Nguyen Tang Cuong was born in 1950, Ass. Professor, Ph.D, Lecture in Le Quy Don Technical University. He received University Education, Ph.D Degree and Post-graduate Technology Practics from Minsk Air Defence Missile Engineering University, and Bauman Moscow State Technical University. He published more than 80 scientific papers in technical related filds of Automation, Flight Equipment Control, Computer System, Signal Processing and Industrial Control System.

Hoàng Đức Trọng sinh năm 1992, tốt nghiệp chuyên ngành Kỹ thuật Điều khiển và Tự động hóa, Học Viện Kỹ Thuật Quân Sự năm 2015. Hiện đang làm việc cho công ty FANUC Việt Nam, là công ty con của Tập Đoàn FANUC Nhật Bản, chuyên về Robot công nghiệp, máy CNC, Roboshot và Robocut dùng trong công nghiệp. Hướng nghiên cứu chuyên sâu về Điều khiển, máy công nghiệp và mạng truyền thông công nghiệp.

Vũ Đức Tân sinh năm 1981, nhận bằng kỹ sư ngành Kỹ thuật hàng không, chuyên ngành Thiết bị hàng không năm 2005. Từ năm 2005 đến năm 2013 công tác tại: Nhà máy A45, Cục Kỹ thuật, Quân chủng Phòng không - Không quân với chức vụ Trợ lý kỹ thuật nhà máy. Hiện nay đang là học viên Lớp cao học kỹ thuật khóa 18, Học viện Phòng không – Không quân.