BÁO CÁO THỰC HÀNH ĐỒ ÁN MÔN HỌC

Thực hiện Service $27H (Security Access) của chuẩn giao tiếp CAN

Môn học: **CHUYÊN ĐỀ THIẾT KẾ HỆ THỐNG NHÚNG 1**- Mã lớp: **CE437.N11**

Giảng viên hướng dẫn thực hành: Phạm Minh Quân

|  |  |
| --- | --- |
| **Thông tin sinh viên** | Mã số sinh viên: 19521022 – 19521387  Họ và tên: Nguyễn Văn Tín – Cao Phan Tiến Dũng |
| **Link các tài liệu tham khảo** *(nếu có)* |  |
| **Đánh giá của giảng viên**:  *+ Nhận xét*  *+ Các lỗi trong chương trình*  *+ Gợi ý* |  |

*[Báo cáo chi tiết các thao tác, quy trình sinh viên đã thực hiện trong quá trình làm bài thực hành. Chụp lại hình ảnh màn hình hoặc hình ảnh kết quả chạy trên sản phẩm. Mô tả và giải thích chương trình tương ứng để cho ra kết quả như hình ảnh đã trình bày. Sinh viên xuất ra file .pdf và đặt tên theo cấu trúc: MSSV\_HoTen\_Labx\_Report.pdf (Trong đó: MSSV là mã số sinh viên, HoTen là họ và tên, x trong Labx là chỉ số của bài thực hành tương ứng]*

Mục lục

[1 Tổng quan hệ thống: 3](#_Toc124176904)

[1.1 Dịch vụ Security Access 3](#_Toc124176905)

[1.2 Cấu trúc gói tin 4](#_Toc124176906)

[1.2.1 Request Seed 4](#_Toc124176907)

[1.2.2 Send Key 5](#_Toc124176908)

[2 Hiện thực một số hàm ở lớp mạng 5](#_Toc124176909)

[2.1 Các cấu trúc dữ liệu 5](#_Toc124176910)

[2.1.1 CanTP\_HandleTypedef 5](#_Toc124176911)

[2.1.2 CanTP\_Packet 6](#_Toc124176912)

[2.1.3 Khác 6](#_Toc124176913)

[2.2 Các hàm hiện thực 6](#_Toc124176914)

[2.2.1 Hàm khởi tạo 6](#_Toc124176915)

[2.2.2 Hàm gửi CanTP\_Transmit 7](#_Toc124176916)

[2.2.3 Hàm nhận CanTP\_Receive 9](#_Toc124176917)

[3 Hiện thực dịch vụ $27H Security Access ở lớp chẩn đoán: 10](#_Toc124176918)

[3.1 Xử lý trên ECU 10](#_Toc124176919)

[3.1.1 Khởi tạo 10](#_Toc124176920)

[3.1.2 Vòng lặp 12](#_Toc124176921)

[3.2 Xử lý trên Tester 13](#_Toc124176922)

[3.2.1 Khởi tạo 14](#_Toc124176923)

[3.2.2 Kịch bản kiểm tra. 15](#_Toc124176924)

[3.2.3 Chương trình xử lý. 15](#_Toc124176925)

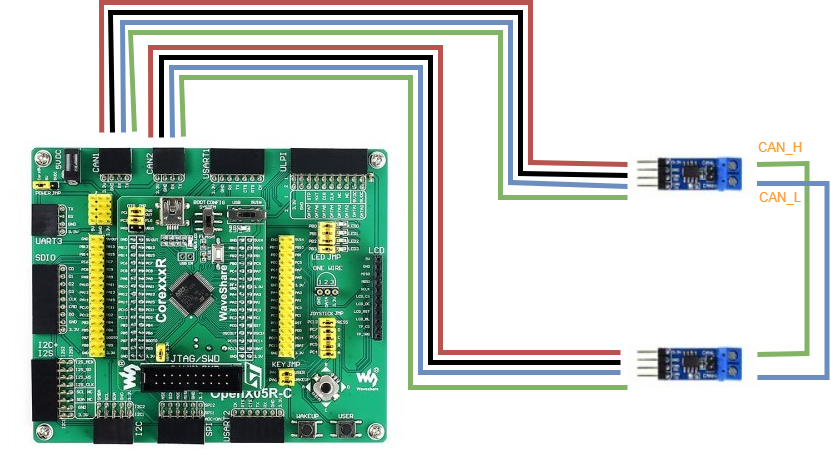
[4 Kết quả 17](#_Toc124176926)

[4.1 Trường hợp gửi Tester gửi lại Key sai. 17](#_Toc124176927)

[4.2 Trường hợp Tester tính và gửi key đúng. 18](#_Toc124176928)

# Tổng quan hệ thống:

Trên KIT thực hành có 2 module CAN1 và CAN2, ta thực hiện kết nối mạng CAN giữa 2 module CAN1 và CAN 2 như **Error! Reference source not found.**.



**Hình 1.1: Sơ đồ kết nối**

CAN1 sẽ đóng vai trò là ECU Module và CAN2 sẽ đóng vai trò là Tester Module. Mỗi module sẽ được chạy ở một luồng riêng. Hai module thực hiện giao tiếp chẩn đoán qua giao thức CAN. Hình 2 mô tả bố trí các lớp trong giao thức CAN.

A picture containing text

Description automatically generated

**Hình 1.2: Sơ đồ các tầng**

Đặc tả của các lớp đã được nêu trong bài thực hành số 4. Đề tài này nhóm sẽ hiện thực service $27H Security Access trong lớp chẩn đoán. Lớp mạng được hiện thực sẽ bao gồm lớp liên kết dữ liệu bên trong. Và thư viện STM32 HAL CAN sẽ được sử dụng để tương tác với lớp vật lý trong module.

Vì ECU và Tester module đều trên cùng một board và ở trên cùng một vii điều khiển nên nhóm sẽ sử dụng thư CMSIS-RTOS v2 để tạo 2 luồng song song một luồng là ECU một luồng là Tester. Hai luồng chỉ có thể giao tiếp với nhau thông qua giao thức CAN như **Error! Reference source not found.**.

## Dịch vụ Security Access

Dịch vụ Security Access được cung cấp như là một phương tiện để truy cập vào dữ liệu và các dịch vụ chẩn đoán mà giới hạn truy cập vì lí do bảo mật, an toàn,…

Dịch vụ Security Access sử dụng cơ chế thuật toán seed và key. Đầu tiên Tester sẽ yêu cầu đến ECU mở khóa qua dịch vụ Security Access – Request Seed. Và sau đó ECU sẽ trả về seed. Từ giá trị seed Tester và ECU sẽ tính toán ra giá trị Key.

Bước tiếp theo, Tester sẽ gửi giá trị Key mà mình tính được về cho ECU thông qua tin nhắn Security Access – Send Key. ECU sẽ so sánh giá trị Key mình nhận được với giá trị đã được tính sẵn trong ECU. Nếu 2 giá trị là trùng nhau, ECU sẽ mở khóa và một số tính năng dịch vụ đặc biệt và thông báo về cho Tester. Còn nếu 2 giá trị không trùng nhau thì sẽ được xem là một lần truy cập lỗi.

Diagram

Description automatically generated

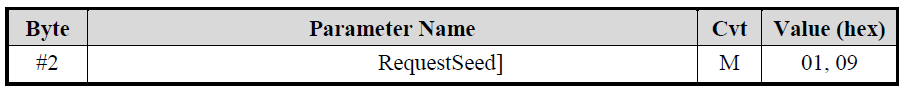
**Hình 1.3: Sơ đồ giao tiếp**

## Cấu trúc gói tin

### Request Seed

Table

Description automatically generated



Graphical user interface, text, application

Description automatically generated

### Send Key

A picture containing text

Description automatically generated

Graphical user interface, application

Description automatically generated

Table

Description automatically generated

# Hiện thực một số hàm ở lớp mạng

## Các cấu trúc dữ liệu

### CanTP\_HandleTypedef

Text

Description automatically generated

Hình 2. Cấu trúc quản lý đối tượng lớp mạng

Trong đó:

* CanTxID: ID gói tin sẽ gửi đến mạng CAN
* CanRxID: Can ID đối tượng sẽ lắng nghe và nhận dự liệu
* Hcan: Đối tượng quản lý việc giao tiếp tầng vật lý của CAN
* CanRxFifo: Fifo nhận dữ liệu gói tin CAN trong phần cứng STM32
* CanFilter: Chứa cấu hình bộ lọc gói tin cho chip STM32
* Status: Cờ báo trạng thái hiện tại của đối tượng.
* CanRxBuffer: Buffer chứa dữ liệu mỗi frame ở tầng vật lý nhận đc. (Vì ở tầng vật lý với gói tin chuẩn giao tiếp CAN chỉ gửi được tối đa 8 Bytes)
* CanRxLen: Số lượng bytes nhận được ở tầng vật lý.

### CanTP\_Packet

A picture containing icon

Description automatically generated

Hình 2. Cấu trúc chứa gói tin nhận và gửi

Trong đó

* Data: Dữ liệu nhận được hoặc sẽ gửi đi.
* Len: Độ dài dữ liệu sẽ gửi đi hoặc được nhận vào.

### Khác

Text

Description automatically generated

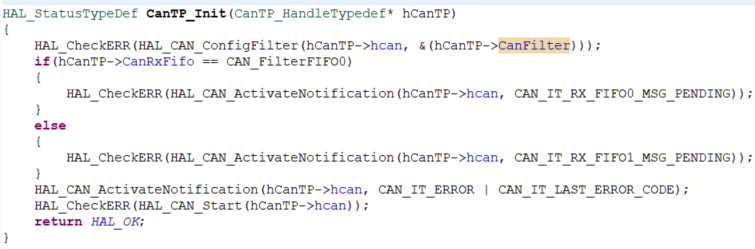
**Hình 2.3: Cấu trúc các khung truyền**

Ngoài ra dựa vào đặc tả định dàng từng gói tin nhóm định nghĩa các cấu trúc sau nhằm dễ dàng quản lý dữ liệu.

* CanTP\_SFFrameTypedef: Cấu trúc chứa dữ liệu gói tin SingleFrame
* CanTP\_FFFrameTypedef: Cấu trúc chứa dữ liệu gói tin FirstFrame
* CanTP\_CFFrameTypedef: Cấu trúc chứa dữ liệu gói tin ConsequenceFrame
* CanTP\_FCFrameTypedef: Cấu trúc chứa dữ liệu gói tin FlowControl

## Các hàm hiện thực

### Hàm khởi tạo



Đầu tiên ta gọi đến hàm cầu hình bộ lọc nhằm mục địch cho vi xử lý biết ta sẽ nghe và nhận những CAN ID nào. Tiếp đó ta tiến hành bật ngắt nhận thông qua hàm HAL\_CAN\_ActivateNotification. Khi có dữ liệu đến FIFO sẽ xảy ra ngắt. Kế đến ta khởi động ngoại vi CAN trên vi điều khiển qua hàm HAL\_CAN\_Start.

### Hàm gửi CanTP\_Transmit

Graphical user interface, text, application, email

Description automatically generated

Hàm gửi ở tầng mạng sẽ bao gồm 2 hàm là hàm TxSingleFrame sẽ gửi đi gói tin SF nếu Data < 7 Bytes. Và hàm TxMultiFrame sẽ cắt gói tin ra thành nhiều gói tin nhỏ và gửi đi theo sơ đồ hình dưới nếu độ dài gói tin lớn hơn 7 Bytes.

Table

Description automatically generated

**Hình 2.4: Bảng cấu trục các gói tin mạng CAN**

Hàm gửi 1 Frame CanTP\_TxSingleFrame:

Text

Description automatically generated with medium confidence

Text

Description automatically generated

Hàm CanTP\_TxSF làm nhiệm vụ đóng gói lại các dữ liệu theo format của khung truyền loại SF sau đó gọi đến hàm CAN\_Trasmit để gửi dữ liệu với ID gói tin là CanTxID. Vì dựa theo đặc tả của lớp liên kết dữ liệu mỗi khung truyền trên mạng CAN sẽ truyền 8 bytes dữ liệu, những byte trống không sử dụng sẽ có giá trị là 0x55.

Vì ở lớp vật lý với mỗi gói tin CAN chỉ gửi được tối đa 8 bytes dữ liệu nên sẽ cần cơ chế cắt ghép các gói tin có kích thước lớn ra thành nhiều gói tin nhỏ để gửi. Quy trình gửi như hình dưới.

Chart, diagram, line chart

Description automatically generated

**Hình 2.5: Sơ đồ giao tiếp mạng CAN ở lớp mạng**

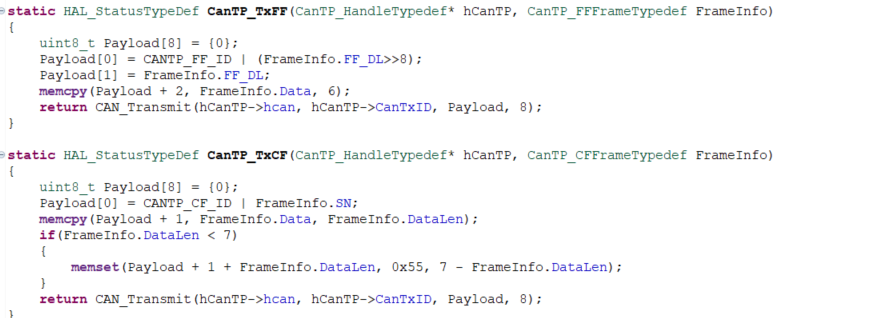
Hàm CanTP\_TxMultiFrame làm nhiệm vụ chia nhỏ các gói tin ra và gửi đi.

Graphical user interface, text, application, email

Description automatically generated

Trước khi gửi dữ liệu trên mạng CAN ta sẽ có một khoảng Delay nhỏ bởi vì ta đang truyền và nhận trên cùng một board với 2 luồng song song do đó cần khoảng delay để nhường tài nguyên cho luồng nhận đảm bảo luồng nhận đã sẵn sàng để nhận.

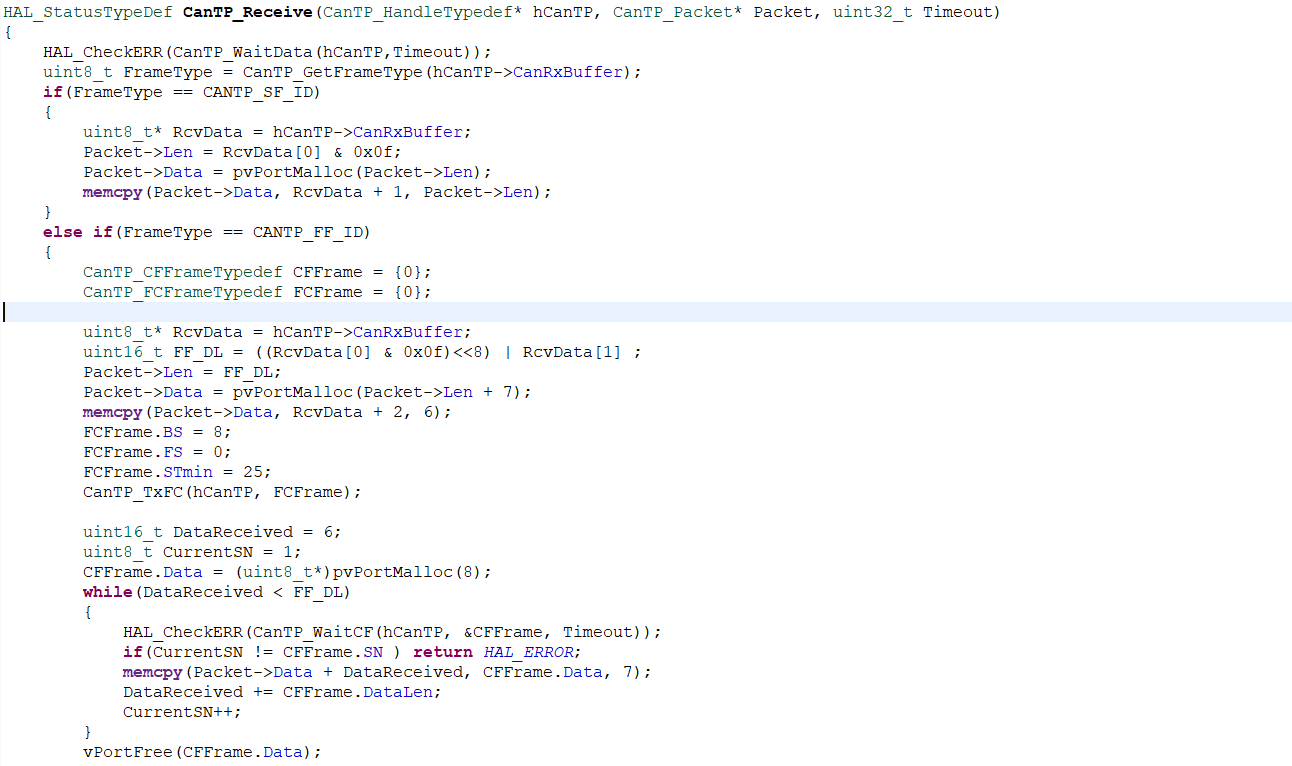
* Đầu tiên ta sẽ gửi 6 bytes đầu của dữ liệu thông qua khung truyền FF.
* Sau đó ta đợi bên nhận gửi một số thông tin cấu hình thông qua khung truyền FC.
* Sau khi nhận được khung truyền FC ta tiến hành gửi tiếp những dữ liệu còn lại thông qua khung truyền CF.



Graphical user interface, text, application

Description automatically generated

### Hàm nhận CanTP\_Receive



Text, letter

Description automatically generated

Nhóm thiết kế hàm nhận theo cơ chế Polling. Trong hàm nhận đầu tiên chương trình sẽ đợi khung dữ liệu đầu tiên được gửi đến.

* Nếu khung dữ liệu đầu thuộc loại SF ta chỉ cần lấy dữ liệu ra và kết thúc hàm nhận
* Nếu khung dữ liệu đầu tiên nhân được thuộc loại FF ta tiến hành gửi tín hiệu FC và tiến vào chu trình nhận dữ liệu đến khi đủ dữ liệu được thông báo trong khung truyền FF.
* Nếu không thuộc hai loại trên hàm sẽ trả về lỗi.

Text

Description automatically generated

Để đợi dữ liệu nhận vào nhóm sử dụng cơ chế EventFlagWait của RTOS kết hợp với ngắt nhận của CAN trong STM32.

Khi có dữ liệu đến sẽ gọi đến hàm ngắt RcvCBHandler. Trong này ta sẽ tiến hành đọc các dữ liệu nhận được và phát một cờ tín hiệu để luồng đang nhận dữ liệu được mở khóa và xử lý các tín hiệu đó.

# Hiện thực dịch vụ $27H Security Access ở lớp chẩn đoán:

## Xử lý trên ECU

Graphical user interface

Description automatically generated

### Khởi tạo

Text, letter

Description automatically generated

Text

Description automatically generated

Ta khởi tạo một đối tượng ECUCanTP để quản lý quá trình giao tiếp truyền nhận dữ liệu theo lớp mạng. Với các cấu hình như sau. ECU sẽ sử dụng CAN1 trên vi điều khiển để giao tiếp. Sử dụng Fifo0 trong CAN1 để nhận các gói tin. Các gói tin gửi đi trong mạng CAN sẽ có ID là 0x712 và sẽ nhận và xử lý những gói tin có ID là 0x7A2.

Graphical user interface, text, application, email

Description automatically generated

Ngoài ra ta cài đặt hàm ngắt Callback xử lý các gói tin nhận được của ECUCanTP.

### Vòng lặp

Graphical user interface, text, application, email

Description automatically generated

Chương trình sẽ tiến hành đợi dữ liệu đến. Khi có dữ liệu đến ta lấy ra Bytes đầu tiên của gói tin để kiểm tra gói tin đó là dịch vụ gì nếu là dịch vụ SecurityAccess $27H sẽ gọi đến hàm ECU\_SecurityAccess\_Service nhắm vào chu trình thực hiện dịch vụ đó.

Text

Description automatically generated

Với SubFuncID là 0x01 sẽ là lệnh RequestSeed SecurityAccess Service. Khi đó ta sẽ sinh ra một số để làm seed cho thuật toán mã hóa Seed-Key. Để sinh ra seed nhóm sẽ sử dụng ngoại vi RNG (Random Number Generator) để tạo một số ngẫu nhiên 32bit là seed cho dịch vụ Security Access.

Sau khi tính toán seed sẽ phản hồi về Tester.

Graphical user interface, text, application, email

Description automatically generated

Từ giá trị seed vừa có được sẽ ECU sẽ tính toán ra Key. Ở đây nhóm sử dụng thuật toán crc32 để sinh Key. Sau khi có Key ECU sẽ tiến hành đợi trong vòng 500ms để đợi gói tin Security Access – SendKey từ Tester.

Graphical user interface, text, application, email

Description automatically generated

Sau khi nhận được gói tin SendKey ECU sẽ tiến hành kiểm tra giá trị Key mình nhận được và giá trị Key mình tính được có trùng nhau hay không. Nếu trùng nhau sẽ tiến hay mở khóa ECU và gửi về gói tin phần hồi thành công đến Tester. Nếu khác sẽ tính là truy cập sai và thực hiện việc khóa ECU

Graphical user interface, text, application

Description automatically generated

Nhóm sẽ thực hiện việc bật và tắt LED1 trên board thực hành để mô tả trạng thái của board là có khóa hay không. Nếu LED1 được bật tức ECU đang được mở khóa và tắt nếu ECU đang ở trạng thái khóa.

## Xử lý trên Tester

Graphical user interface, text, application

Description automatically generated

### Khởi tạo

Text

Description automatically generated

Text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Tương tự như khi ta khởi tạo ECU. Ta sẽ khởi tạo Tester sẽ sử dụng CAN2 trên vi điều khiển để giao tiếp. Dùng FIFO1 để đọc và nhận gói tin. Các gói tin Tester gửi đi sẽ có ID là 0x7A2 và sẽ nhận vào và xử lý các gói tin có ID 0x712.

### Kịch bản kiểm tra.

Sau khi Tester request Seed từ ECU, Tester sẽ tính toán ra giá trị Key.

* Tester gửi Key đúng ECU sẽ được mở khóa và đèn LED1 sẽ sáng.
* Tester gửi Key sai ECU sẽ bị khóa lại đèn LED1 không sáng.

### Chương trình xử lý.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application

Description automatically generated

Nhóm sử dụng Interrupt để đọc các nút nhấn. Khi nhấn nút sẽ gọi đến hàm xử lý ngắt. Đối ứng với nút ta nhấn sẽ bật cờ báo đã nhấn nút đó lên.

Bắt đầu chu trình ta sẽ nhấn nút 0 (WakeupBTN trên kit TH) bắt đầu chương trình kiểm tra dịch vụ Security Access.

Nếu ta tiếp tục nhấn nút 0 một lần nữa Tester sẽ cố tình gửi một Key sai ECU.

Nếu ta nhấn nút 1 (USER button trên kit TH) chương trình sẽ tính toán Key dựa trên Seed và gửi đến ECU.

Graphical user interface, text, application, email

Description automatically generated

Để bắt đầu dịch vụ Security Access ta sẽ gửi gói tin SecurityAccess – Request Seed đến cho ECU. Sau đó ta tiến hành đợi gói tin phản hồi từ ECU về trong gói tin sẽ chứa SecuritySeed.

Graphical user interface, text, application, email

Description automatically generated

Sau khi có Seed ECU sẽ tiến hành tính toán ra Key cũng theo thuật toán CRC32 giống ECU. Tiếp đến Tester gửi giá trị Key mình tính toán được đến cho ECU kiểm tra thông qua dịch vụ Security Access – SendKey.

Graphical user interface, text, application, email

Description automatically generated

# Kết quả

## Trường hợp gửi Tester gửi lại Key sai.

Graphical user interface, text, application, email

Description automatically generated

**Hình 4.1: Trường hợp Tester gửi Key sai**

Tester gửi gói tin 0x27 0x01 tức Security Access – Request Seed đến ECU và ECU báo đã nhận được gói tin và tiến hành vào dịch vụ Security Access. ECU sinh ra seed 1139721040 và tính được Key là -869933076.

* Tester báo là nhận được Seed với giá trị 1139721040 và vì ta đang để Tester gửi Key sai nên sẽ gửi về ECU key là 123456789.
* ECU nhận được Key là 123456789 khác với key ECU tính được nên ECU sẽ bị khóa.

## Trường hợp Tester tính và gửi key đúng.

Graphical user interface, text

Description automatically generated

**Hình 4.2: Trường hợp Tester gửi key đúng**

Tương tự như trên:

* Tester gửi đến ECU gói tin Security Access – Request Seed.
* ECU sinh ra seed với giá trị 764492380
* ECU tính ra Key với giá trị -1495440576
* Tester nhận được seed với giá trị 764492380
* Tester tính toán và gửi key đến ECU với giá trị -1495440576
* Vì giá trị Key ECU tính toán được và giá trị ECU nhận được giống nhau nên ECU tiến hành mở khóa. Và phản hổi gói tin mở khóa thành công đến Tester.
* Tester nhận được gói tin phản hồi mở khóa thành công.