Mục lục

[1. Giới thiệu 1](#_Toc111015903)

[2. PKCS#11 là gì? 1](#_Toc111015904)

[3. Nhà cung cấp 2](#_Toc111015905)

[3.1. Nhà cung cấp Sun PKCS#11 2](#_Toc111015906)

[3.2. PKCS #11 hoạt động như thế nào? 2](#_Toc111015907)

[4. Ví dụ 4](#_Toc111015908)

[5. Cài SoftHSM trên Ubuntu 5](#_Toc111015909)

[5.1. Setup 5](#_Toc111015910)

[5.2. Cấu hình file 8](#_Toc111015911)

[5.3. Sử dụng SoftHSM 8](#_Toc111015912)

[5.4. Tương tác với token trong Java 9](#_Toc111015913)

[6. Cài SoftHSM trên Window 12](#_Toc111015914)

[6.1. Cài đặt SoftHSM2 bằng MSI Installer 12](#_Toc111015915)

[6.2. Sử dụng SoftHSM 23](#_Toc111015916)

[7. Vận hành 26](#_Toc111015917)

[8. Kết luận 27](#_Toc111015918)

# 1. Giới thiệu

SoftHSM là một triển khai của một kho lưu trữ mật mã có thể truy cập được thông qua giao diện PKCS # 11. Bạn có thể sử dụng nó để khám phá PKCS # 11 mà không cần có HSM. SoftHSM là một phần mềm giả lập của HSM liên quan đến các khóa mật mã. Mục đích của các thiết bị này là tạo khóa mật mã và ký thông tin mà không tiết lộ private key ra bên ngoài. Bên cạnh đó, SoftHSM là dùng để lưu private key hoặc certificate trên file và mã hóa tăng tốc.

<https://www.opendnssec.org/softhsm/>

# 2. PKCS#11 là gì?

Trước tiên, chúng ta phải hiểu về phần “PKCS # 11 là gì?”

PKCS # 11 là một API tiêu chuẩn được chỉ định bởi OASIS Open, một tổ chức phi lợi nhuận toàn cầu hoạt động dựa trên sự phát triển, hội tụ và áp dụng các tiêu chuẩn mở về bảo mật, IoT, năng lượng, công nghệ nội dung, quản lý khẩn cấp và các lĩnh vực khác. Họ xác định tiêu chuẩn PKCS # 11 như sau:

“**Tiêu chuẩn PKCS # 11** chỉ định giao diện lập trình ứng dụng (API), được gọi là“ Cryptoki ”, dành cho các thiết bị lưu trữ cryptographic information và thực hiện các cryptographic functions.” - Tài liệu OASIS

**Tiêu chuẩn PKCS # 11**

PKCS # 11 chỉ định Giao diện lập trình ứng dụng (API) được gọi là Cryptoki cho các thiết bị lưu trữ cryptographic information và thực hiện các cryptographic functions. Những thiết bị này được gọi là Cryptographic Tokens. PKCS # 11 trình bày một quan điểm logic chung về tokens cho các ứng dụng.

<https://clydedcruz.medium.com/a-dive-into-softhsm-e4be3e70c7bc>

Các mục tiêu của PKCS # 11 và cách tiếp cận cơ bản của chúng được xác định như sau,

“Cryptoki tuân theo phương pháp tiếp cận dựa trên đối tượng đơn giản, giải quyết các mục tiêu về độc lập công nghệ (bất kỳ loại thiết bị nào) và chia sẻ tài nguyên (nhiều ứng dụng truy cập vào nhiều thiết bị), trình bày cho các ứng dụng một cái nhìn logic, chung về thiết bị được gọi là“ cryptographic token”. ” - Tài liệu OASIS

<http://docs.oasis-open.org/pkcs11/pkcs11-base/v2.40/os/pkcs11-base-v2.40-os.html>

# 3. Nhà cung cấp

Sau đó đến phần **Cryptography interfaces** trong Java

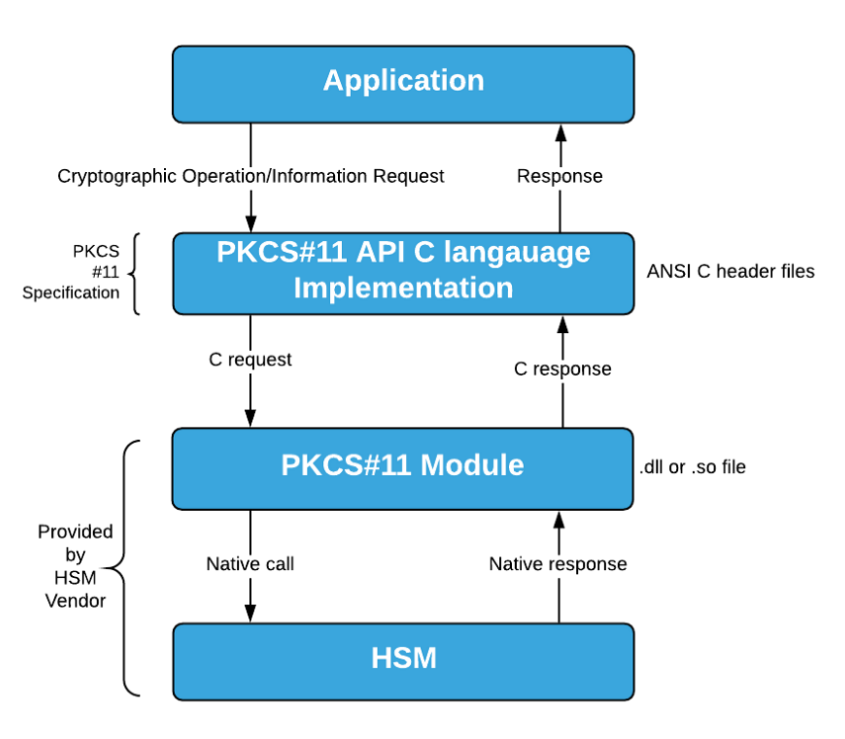
Java định nghĩa một tập hợp các programming interfaces để thực hiện cryptographic operations. Chúng được gọi chung là Java Cryptography Architecture (JCA) và Java Cryptography Extension (JCE). **Các interfaces này dựa trên nhà cung cấp**.

## 3.1. Nhà cung cấp Sun PKCS#11

The Sun PKCS # 11 provider không tự triển khai các thuật toán mật mã mà đóng vai trò là cầu nối giữa các API JCA, JCE và mô-đun PKCS # 11 gốc. Mô-đun gốc này phải ở dạng thư viện đối tượng chia sẻ (tệp .so trên Solaris và Linux) hoặc thư viện liên kết động (.dll trên Windows) và được cung cấp bởi nhà cung cấp thiết bị HSM

<https://clydedcruz.medium.com/a-dive-into-softhsm-e4be3e70c7bc>

## ****3.2. PKCS #11 hoạt động như thế nào?****



Sơ đồ trên cho thấy luồng giao tiếp cấp cao của một ứng dụng với HSM bằng PKCS # 11 API.

Đây là những gì xảy ra trong sơ đồ…

Ứng dụng call API PKCS # 11 và API chịu trách nhiệm gọi mô-đun PKCS # 11 thông qua các lệnh gọi C. Sau đó, mô-đun call tới HSM thông qua các native call. Mô-đun chuyển phản hồi từ HSM đến ứng dụng thông qua việc triển khai giao diện C.

Vì vậy, hãy cùng tìm hiểu sâu hơn và xem cách ứng dụng hoạt động với HSM thông qua PKCS # 11 API. Đầu tiên, có một số từ mà bạn cần phải làm quen:

**Token**Token là logical view của thiết bị mật mã cơ bản. Một token sở hữu danh sách các chức năng mật mã được thiết bị hỗ trợ. ?

**Slot**

Đây là một điểm truy cập hợp lý vào thiết bị mật mã. Các đối tượng nằm trong một vị trí nhất định sẽ không hiển thị với các slot khác.

**Session**

Session là một kết nối logic giữa một ứng dụng và một token. Có hai loại phiên được định nghĩa trong PKCS # 11 là Đọc / Ghi (R / W) và Chỉ đọc (R / O). Phiên R / W có thể được sử dụng cho cả việc đọc và ghi dữ liệu vào thiết bị mật mã trong khi R / O chỉ có thể được sử dụng cho mục đích đọc dữ liệu từ thiết bị.

**User**

User là một người hoặc một ứng dụng có quyền truy cập vào thiết bị mật mã thông qua 1 slot. Về cơ bản, có hai người dùng được định nghĩa trong PKCS # 11 là SO (Nhân viên bảo mật) và USER cho mỗi slot. SO có quyền tạo USER. USER chịu trách nhiệm sử dụng thiết bị cho các hoạt động mật mã. Chỉ có thể có một SO và USER cho một vị trí nhất định.

<https://medium.com/@mevan.karu/standard-api-for-connecting-hsms-with-client-applications-6296eb187d89>

# 4. Ví dụ

Hãy xem điều gì xảy ra giữa ứng dụng và thiết bị mật mã logic bằng cách lấy tình huống sau làm ví dụ.

Crypto là một ứng dụng đang sử dụng HSM được hỗ trợ PKCS # 11 làm nhà cung cấp dịch vụ mật mã. Crypto cần tạo khóa AES bằng HSM và mã hóa một mẫu dữ liệu bằng khóa được tạo.

Chuỗi hành động được thực hiện như sau:

1. Crypto tự xác thực với tư cách là ‘USER’ của người dùng đối với HSM và tạo ra một đoạn giao tiếp an toàn giữa thiết bị và Crypto.
2. Crypto yêu cầu HSM tạo khóa AES thông qua đoạn giao tiếp đã tạo.
3. HSM trả về khóa AES đã tạo.
4. Crypto gửi tập hợp dữ liệu cần được mã hóa bằng khóa mã hóa thông qua đường dẫn an toàn.
5. HSM gửi lại dữ liệu đã được mật mã cho ứng dụng thông qua đoạn giao tiếp.
6. Crypto đóng đoạn giao tiếp.

# 5. Cài SoftHSM trên Ubuntu

Bên trên là ví dụ về 1 luồng trong HSM. Còn dưới đây là cách cài và sử dụng SoftHSM trên **Ubuntu**:

## 5.1. Setup

SoftHSM có thể được cài đặt bằng cách sử dụng binary packages đã có sẵn, hoặc download từ source: <https://wiki.opendnssec.org/display/SoftHSMDOCS/SoftHSM+Documentation+v2>

Ví dụ: trên phiên bản Ubuntu gần đây, một câu lệnh: **sudo apt-get install softhsm** là đủ

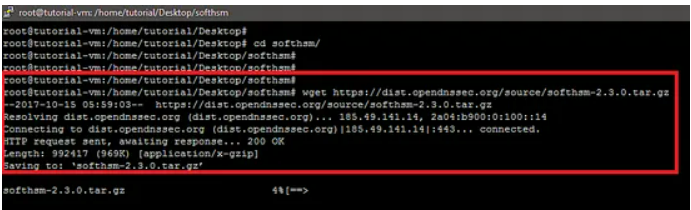
Là một phần của quá trình cài đặt, các công cụ dòng lệnh cần thiết sẽ được cài đặt cùng với tệp đối tượng được chia sẻ **libsofthsm2.so**

Ngoài ra, trên Ubuntu có cách cài khác:

**5.2. Cài đặt**

SoftHSM có sẵn trên trang web OpenDNSSEC và có thể tải xuống bằng lệnh wget theo cách sau.

*wget*[*https://dist.opendnssec.org/source/softhsm-<version>.tar.gz*](https://dist.opendnssec.org/source/softhsm-%3cversion%3e.tar.gz)



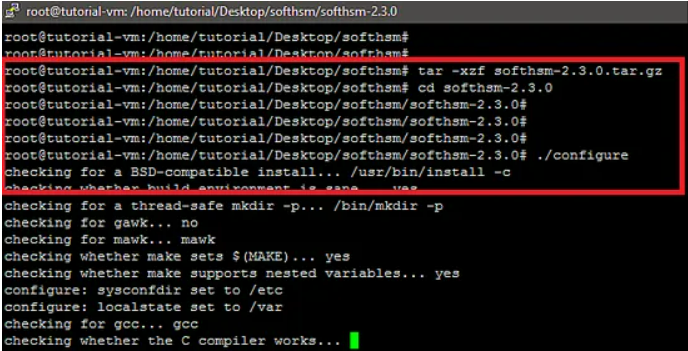
Bây giờ, giải nén gói bằng lệnh tar theo cách sau:

**tar -xzf softhsm-<version>.tar.gz**

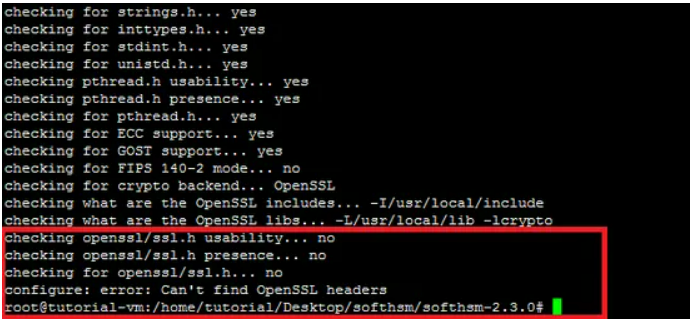
Sau khi giải nén, hãy chạy tập lệnh cấu hình để kiểm tra dependencies của phần mềm SoftHSM:

**cd softhsm-<version>**

**./configure –disable-gost**

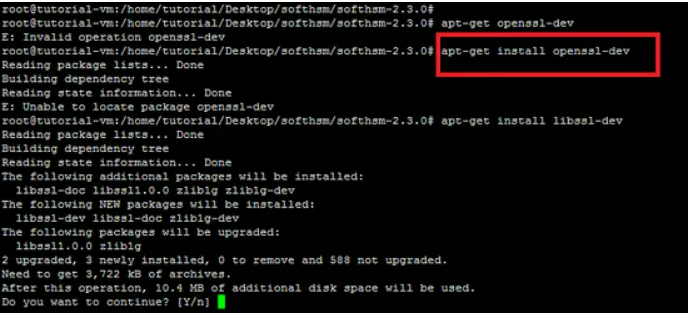
****

Ảnh chụp nhanh sau đây cho thấy rằng tập lệnh cấu hình hiển thị lỗi do thiếu gói "tiêu đề OpenSSL".



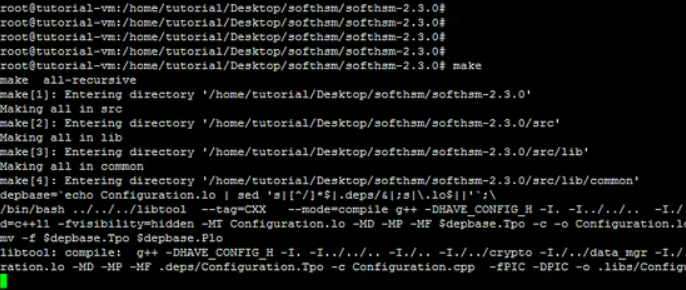
Vì vậy, giải quyết vấn đề này, cài đặt gói "openssl-dev" theo cách được đánh dấu như bên dưới.

***apt-get install openssl-dev/*** ***sudo apt-get install libssl-dev***

****

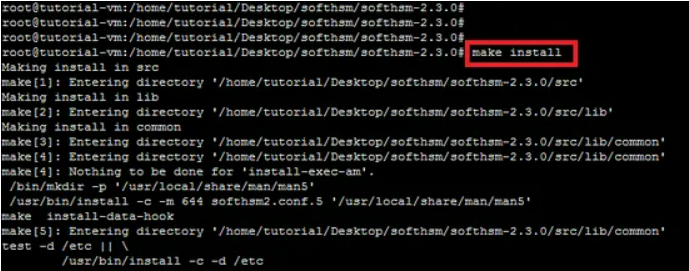
Bây giờ, hãy chạy lệnh make để biên dịch mã nguồn của SoftHSM.

*make*



Tiếp theo, chạy "make install command" để cài đặt công cụ SoftHSM.

*make install*

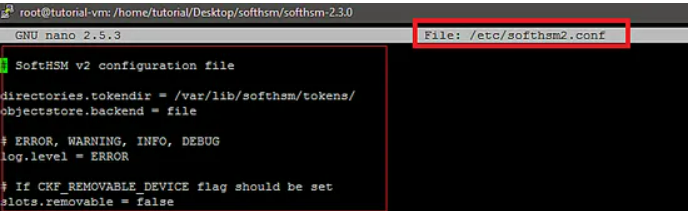


## 5.2. Cấu hình file

Vị trí mặc định của tệp cấu hình là /etc/softhsm2.conf có thể được thay đổi bằng cách đặt biến môi trường SOFTHSM2\_CONF.

*export SOFTHSM2\_CONF=Path\_of\_SoftHSM\_Conf\_file*

Cài đặt mặc định của SoftHSM được hiển thị trong ảnh chụp nhanh sau đây.



<https://www.howtoforge.com/tutorial/how-to-install-and-use-softhsm-on-ubuntu-1604-lts/>

## 5.3. Sử dụng SoftHSM

**Tạo 1 cryptograpic token**

Sử dụng softhsm2-util command line để tạo 1 mã token mới như sau:

softhsm2-util --init-token --slot 0 --label "<token name>"

Thao tác này sẽ nhắc bạn nhập mã PIN SO (được sử dụng để thêm / xóa token) và mã PIN USER (được sử dụng để truy cập token từ ứng dụng).

**Hiện tất cả token**

Sử dụng command line **softhsm2-util --show-slots**

**Kiểm tra tất cả đối tượng bên trong 1 slot**

**pkcs11-tool --modul /usr/local/lib/softhsm/libsofthsm2.so -l -O --slot 1574641475**

**Xóa token**

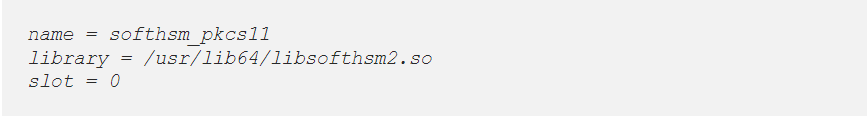
Sử dụng command line **softhsm2-util --delete-token --token “label”**

Nơi lưu trữ token trong Ubuntu **var/lib/tokens/...**

**Tạo 1 PKCS#11 configuration file**

Tệp cấu hình PKCS # 11 phải chứa đường dẫn của tệp .so và slot number của token.

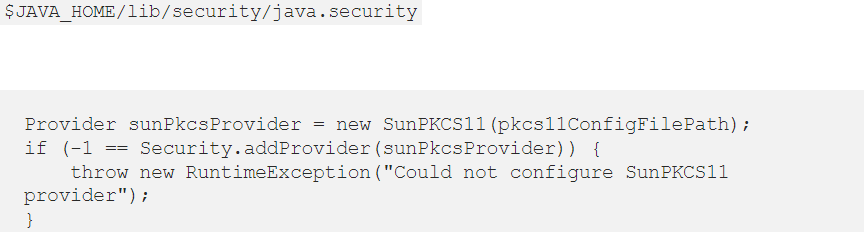
File **libsofthsm2.so** thường được lưu ở usr/local/lib/softhsm



## 5.4. Tương tác với token trong Java

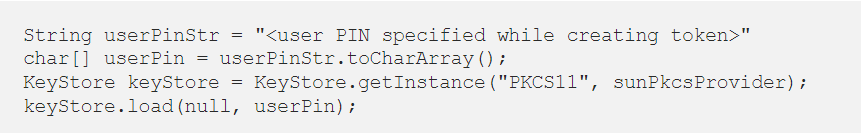
**Cấu hình nhà cung cấp JCA**

Trong đoạn code này, tôi đã lập trình định cấu hình trình cung cấp JCA trong thời gian chạy. Bạn cũng có thể định cấu hình tĩnh nó trong tệp thuộc tính bảo mật Java



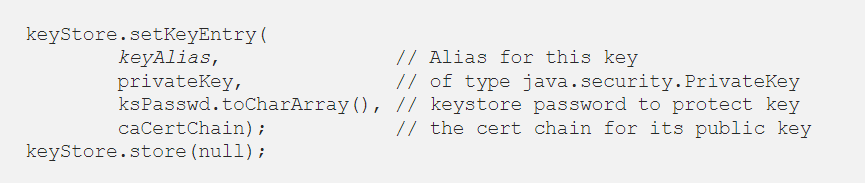
**Load keystore**

Các hoạt động xử lý các keys trong token, yêu cầu bạn đăng nhập trước bằng mã PIN USER của mình trước khi các hoạt động có thể tiếp tục. Bạn có thể cung cấp mã PIN trong tham số nhập mật khẩu cho phương thức tải trong khi truy cập mã thông báo dưới dạng kho khóa thông qua lớp java.security.KeyStore



**Lưu Private key vào keystore**

Ở đây, tôi đang lưu một khóa cá nhân RSA 2048 vào kho khóa

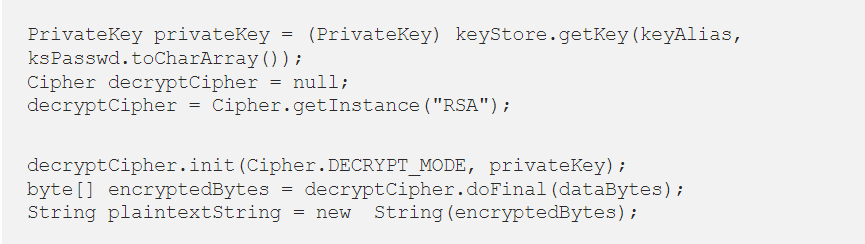


Bạn vừa tạo khóa RSA của mình qua PKCS # 11! Bạn có thể kiểm tra xem nó có trong token của mình hay không bằng lệnh pkcs11-tool

**pkcs11-tool --module /usr/local/lib/softhsm/libsofthsm2.so --list-objects**

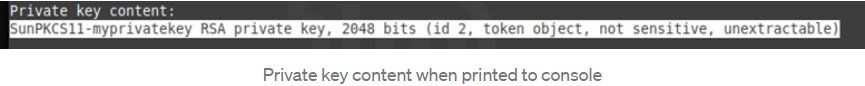
**Các thao tác sử dụng khóa được lưu trữ**

Bây giờ chúng ta có thể sử dụng **key được lưu trữ** để thực hiện các **cryptographic operations**. Trong trường hợp này, tôi đang sử dụng **khóa** để giải mã nội dung đã được mã hóa bằng **khóa công khai tương ứng.**



Điều gì sẽ xảy ra nếu chúng tôi in private key trên consolde?

HSM/ SoftHSM sẽ cho phép bạn thực hiện các hoạt động crypto operations bằng cách sử dụng private key được lưu trữ, nhưng sẽ không cho phép bạn truy cập vào nội dung của chính khóa đó. Đây là những gì chúng tôi nhận được nếu chúng tôi cố gắng in khóa vào bảng điều khiển thông qua Java code.



# 6. Cài SoftHSM trên Window

Đối với windows, chúng tôi có thể tải xuống gói từ: [**https://github.com/disig/SoftHSM2-for-Windows**](https://github.com/disig/SoftHSM2-for-Windows)



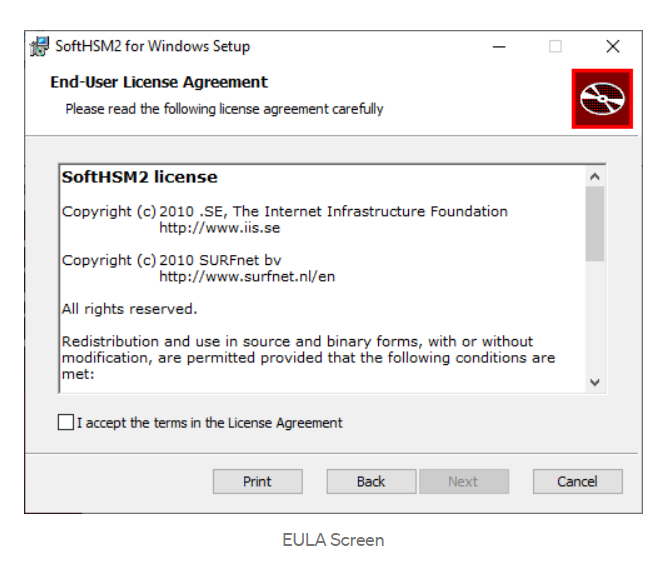
Chọn trình cài đặt portable hoặc MSI Installer. Nếu chúng ta chọn portable, cần thiết lập / sửa đổi các biến môi trường SOFTHSM2\_CONF và PATH theo cách thủ công.

## 6.1. Cài đặt SoftHSM2 bằng MSI Installer

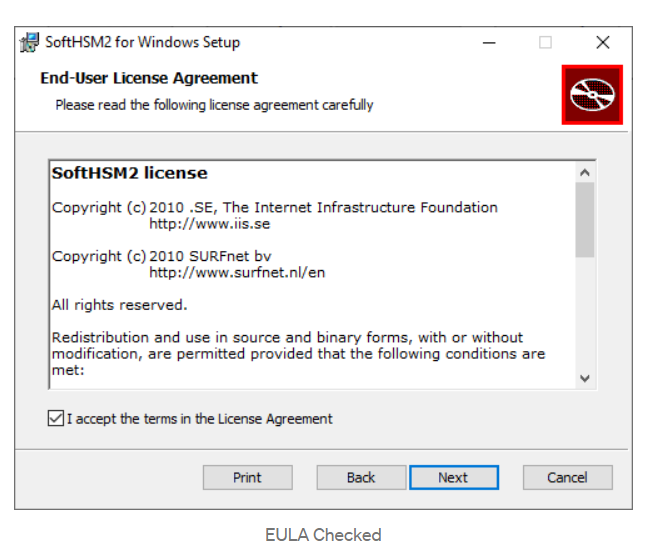
Sau khi tải xuống gói dưới dạng MSI Installer, hãy nhấp đúp vào gói này cho đến màn hình chào mừng



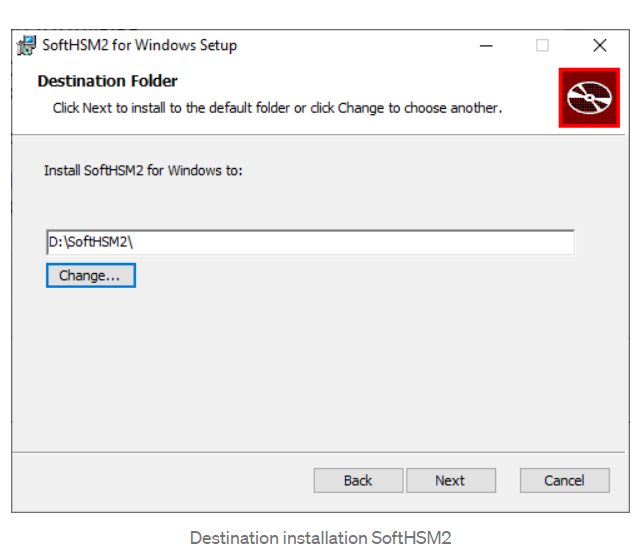
Chọn nút ***Next***



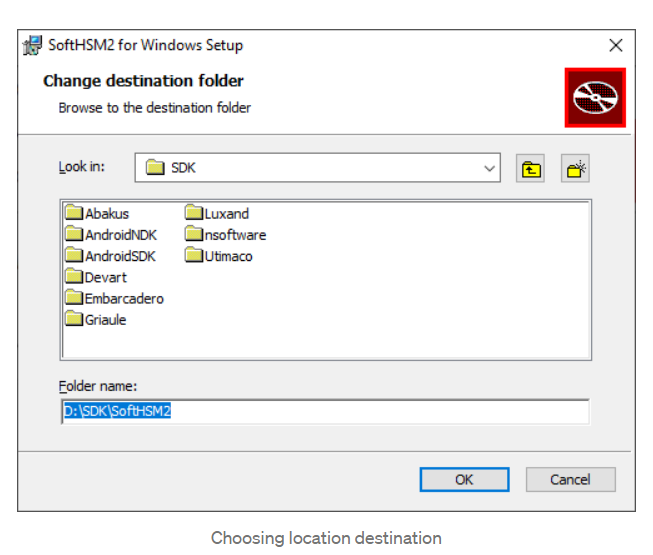
Kiểm tra ***EULA*** và nút ***Next*** sẽ hiển thị

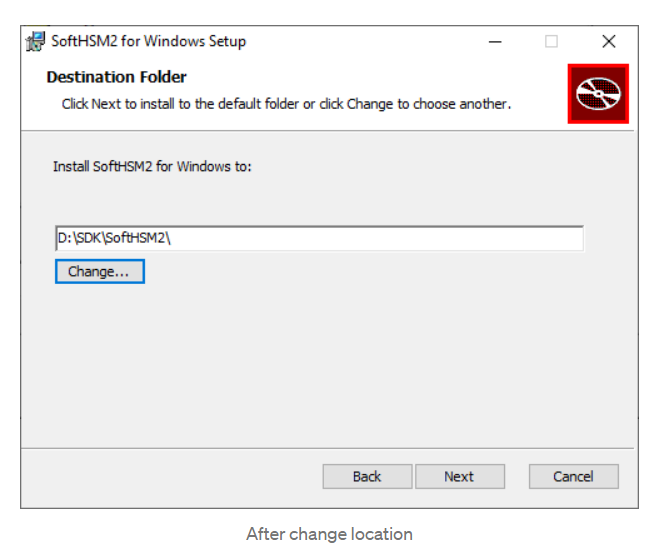


Chọn vị trí cài đặt. Vị trí mặc định D: \ SoftHSM2

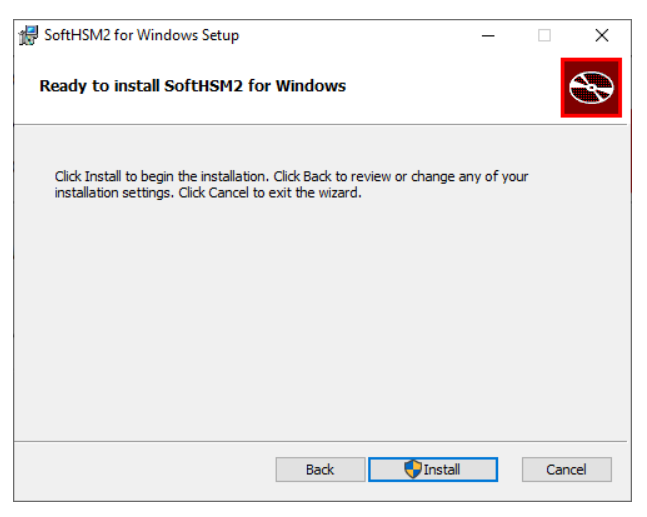


hoặc chúng ta có thể chọn một vị trí khác (nhấn nút Change)

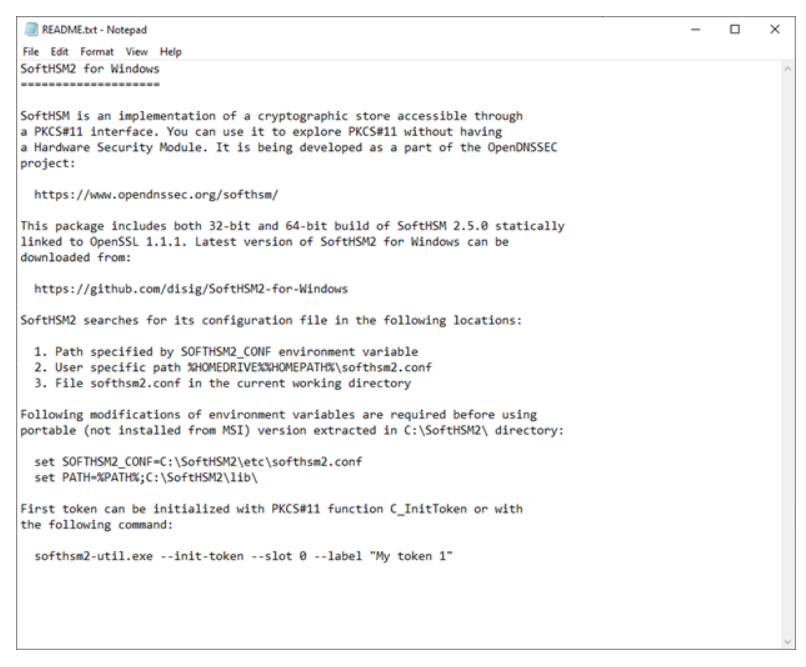




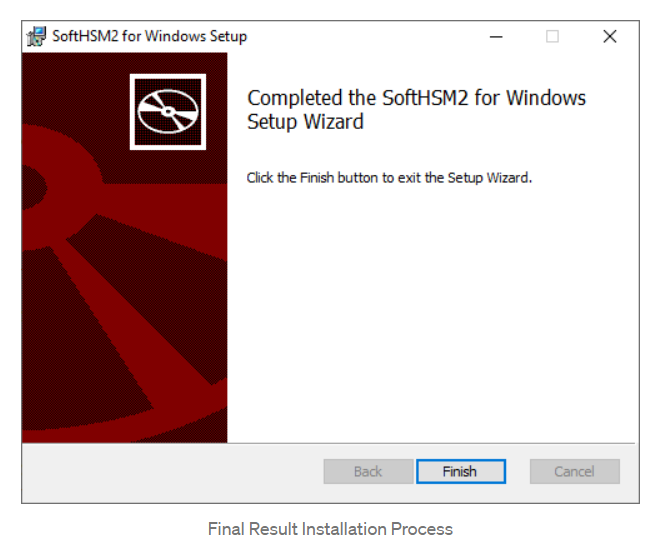
Sau đó nhấn nút ***Next*** rồi nhấn nút ***Install*** để bắt đầu quá trình cài đặt



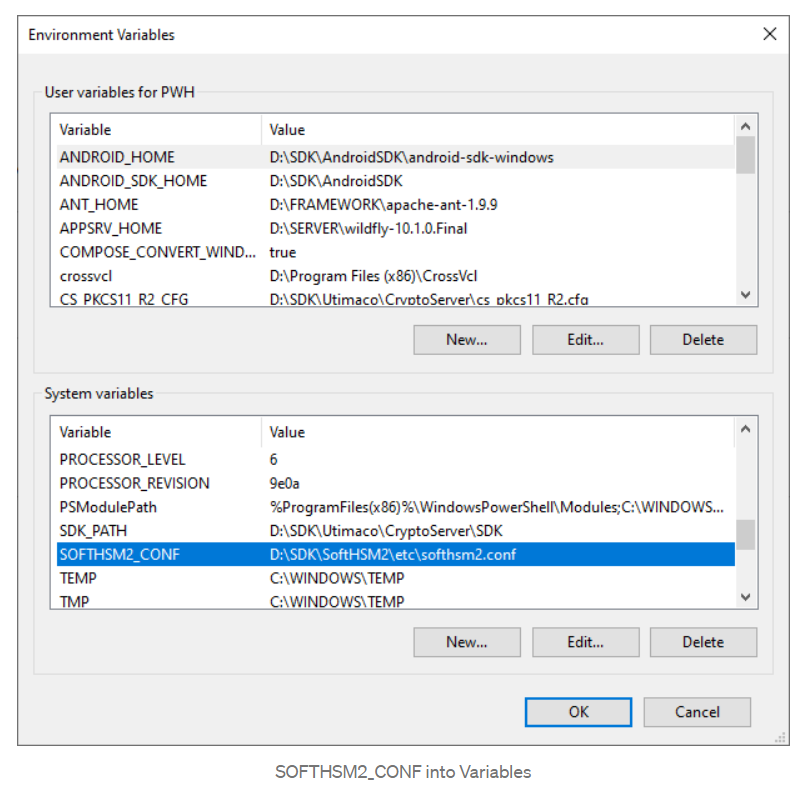
Sau khi quá trình cài đặt hoàn tất sẽ bật lên **README.txt**

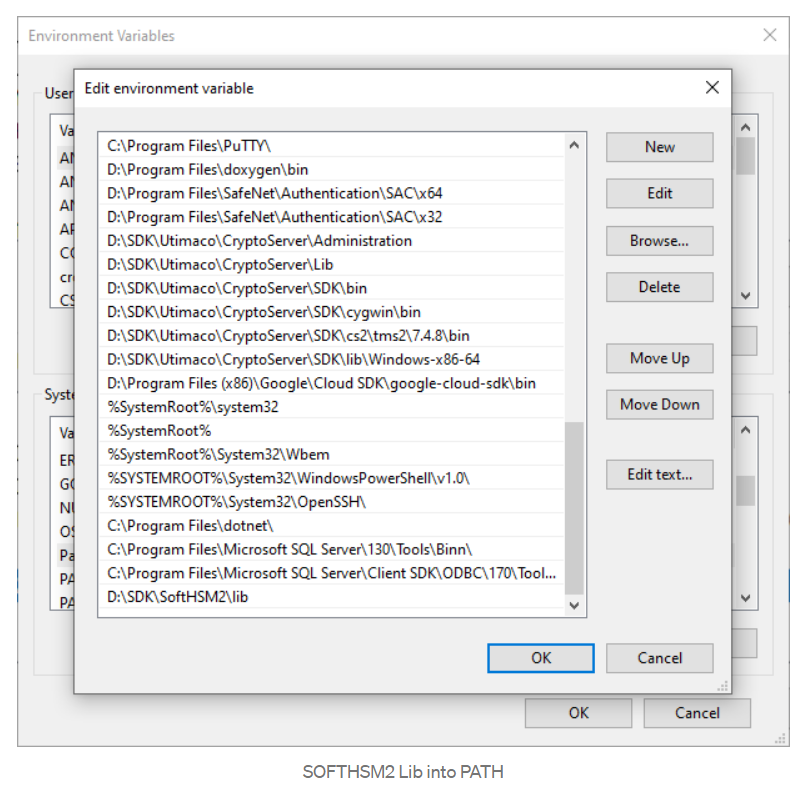


Quá trình cài đặt hoàn tất.



SoftHSM2 sẽ chạy thích hợp là PATH và SOFTHSM2\_CONF thiết lập thành biến môi trường:

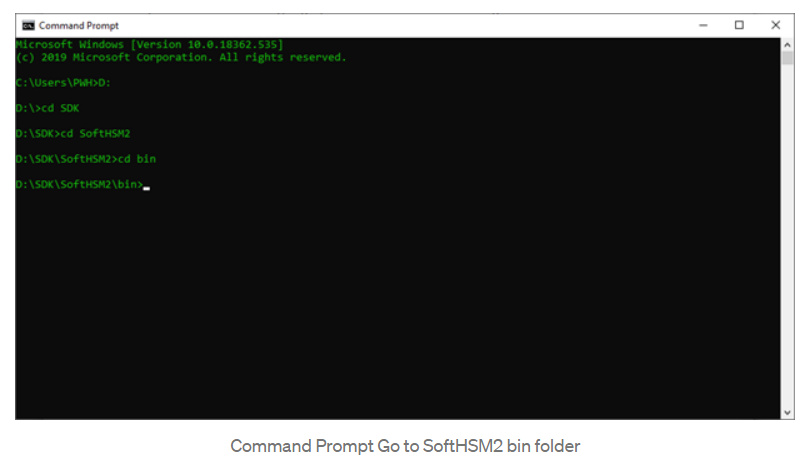




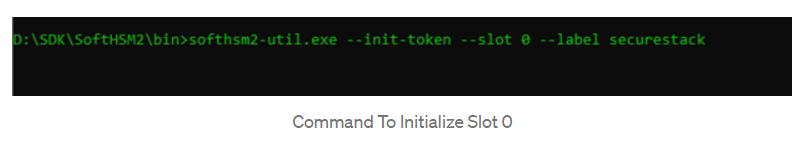
## 6.2. Sử dụng SoftHSM

Sau khi cài đặt xong, chúng ta cần khởi tạo Slots. Kiến trúc SoftHSM2 chia HSM thành nhiều Slots. Mỗi vị trí có một nhà cung cấp bảo mật liên quan và có thể được bảo vệ bằng Mã số nhận dạng cá nhân (PIN) riêng biệt. Bạn có thể tạo một Slot riêng biệt trên SoftHSM2 cho Ứng dụng và bảo vệ Slot này bằng một mã PIN duy nhất.

Mở Command Prompt (cmd.exe) và đi tới thư mục SoftHSM2 \ bin:

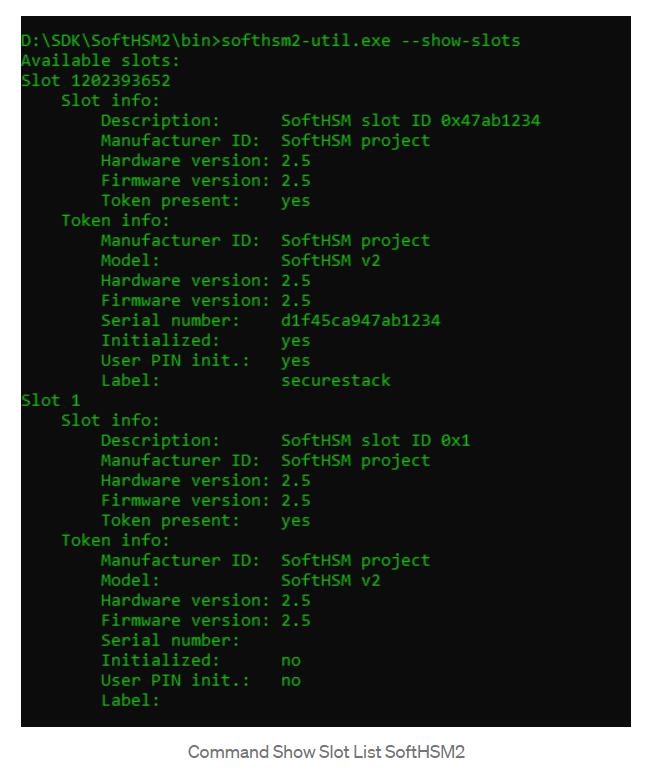


Để khởi tạo Slot 0, chúng ta có thể thực hiện lệnh:



securestack: nhãn slot. Nếu cần, hãy thêm nhãn với dấu cách tách biệt, ví dụ: Nhãn 1 của tôi, chúng ta phải thêm dấu ngoặc kép bên trong nhãn cũ. “My label 1”

Hiển thị danh sách các Slots được tạo trong SoftHSM2:



<https://medium.com/@purnomowhy.1981/installation-guide-softhsm2-in-windows-2e08649411c3>

# 7. Vận hành

Muốn deploy lên docker ứng dụng Java sử dụng SoftHSM để thực hiện các chức năng mã hóa, chúng ta cần có ứng dụng Dockerfile. Trong Dockerfile cần bao gồm: hệ điều hành Ubuntu, SoftHSM, file App.jar, JDK. Bên cạnh đó, chúng ta cần EXPOSE [PORT] và chỉ thị WORKDIR dùng để khai báo thư mục làm việc cho lệnh chỉ thị:CMD.

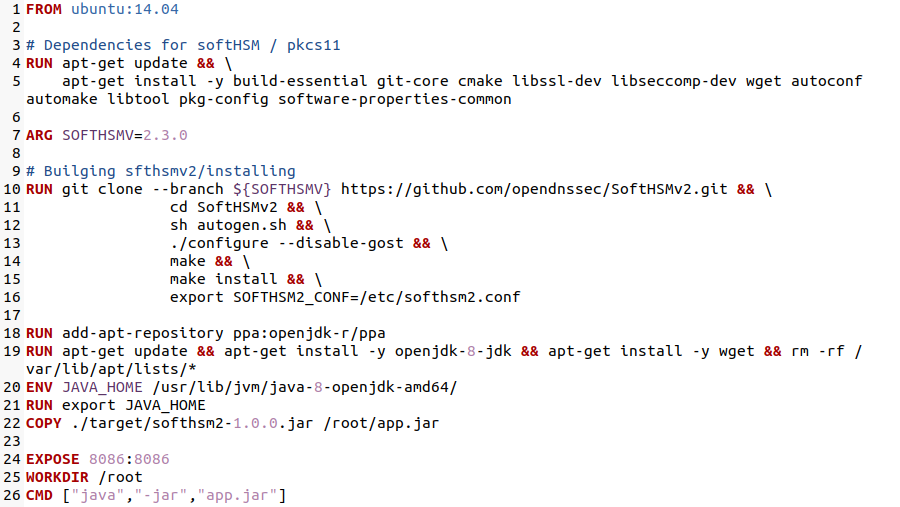


Figure 1*.* ***Dockerfile***

- source code: <http://10.22.18.124:8081/rnd/soft-hsm>

Dùng source code để build thành file jar

- Docker file (được gửi theo file đính kèm)

Ngoài ra, chúng tôi có thể sử dụng docker-compose như sau:



Figure 2*.* ***docker-compose.yml***

# 8. Kết luận

Là một phần mềm triển khai HSM chung, SoftHSM cho phép chúng tôi khám phá tiêu chuẩn cryptoki và hiểu cách sử dụng của nó ngay cả khi chúng tôi không có HSM thực. Trong mã ứng dụng Java, chúng ta có thể tận dụng các API JCA để tương tác với HSM thông qua nhà cung cấp Sun PKCS # 11, nhà cung cấp này hoạt động như một cầu nối giữa các API JCA và mô-đun PKCS # 11 gốc của nền tảng.

<https://clydedcruz.medium.com/a-dive-into-softhsm-e4be3e70c7bc>