Về một số điểm yếu của PKCS#11 và giải pháp khắc phục

Hoàng Văn Thức, Trần Sỹ Nam

***Tóm tắt*— PKCS#11 đã được chấp nhận rộng rãi trong cộng đồng những nhà nghiên cứu và phát triển các thiết bị phần cứng an toàn mật mã, trong đó có thể kể đến các hãng cung cấp các sản phẩm bảo mật và an toàn thông tin như Utimaco, Safenet, Thales, AEP. Trong bài báo này, chúng tôi tổng hợp một số điểm yếu tiềm năng đối với sự an toàn của chuẩn PKCS#11 (phiên bản 2.20), với vai trò là một giao diện lập trình ứng dụng cho một thiết bị phần cứng an toàn mật mã. Trên cơ sở đó, chúng tôi phân tích sự ảnh hưởng và đưa ra các giải pháp cho các nhà phát triển nhằm khắc phục những điểm yếu nêu trên.**

***Abstract*— PKCS # 11 has been widely accepted in the community of researchers and developers of cryptographic security hardware devices, which includes companies providing security and information security products such as Utimaco, Safenet, Thales, AEP. In this article we summarize some potential weaknesses for PKCS # 11 (version 2.20) security, as an application programming interface for a hardware security device. We analyze the impact and provide solutions for developers to overcome the above weaknesses.**

***Từ khóa*— PKCS#11; Thiết bị mật mã an toàn phần cứng, Điểm yếu.**

***Keywords*— PKCS#11; Cryptography security hardware devices, Weakness.**

I. GIỚI THIỆU

Công ty RSA Security đã phát triển và ban hành một bộ các tiêu chuẩn mật mã khóa công khai (Public Key Cryptography Standards - PKCS). Các tiêu chuẩn trong bộ này đề cập đến nhiều khía cạnh trong triển khai ứng dụng mật mã khóa công khai. Ví dụ: tiêu chuẩn mã hóa RSA PKCS#1, tiêu chuẩn mã hóa dựa trên mật khẩu PKCS#5, tiêu chuẩn cú pháp thông tin khóa PKCS#8, tiêu chuẩn giao tiếp thẻ (token) mật mã PKCS#11 [1].

Bài báo được nhận ngày 2/6/2017. Bài báo được gửi cho phản biện thứ nhất vào ngày 17/6/2017 và nhận được ý kiến đồng ý đăng của phản biện thứ nhất đăng vào ngày 9/7/2017. Bài báo được gửi cho phản biện thứ hai vào ngày 27/6/2017 và nhận được ý kiến đồng ý đăng của phản biện thứ hai vào ngày 19/7/2017.

Tiêu đề của chuẩn PKCS#11 đã thể hiện mục tiêu thiết kế và nội dung mà chuẩn đề cập là "*cung cấp một giao diện chuẩn giữa các ứng dụng và các thiết bị mật mã*" và tại cùng thời điểm *"cho phép chia sẻ tài nguyên trong quan hệ lẫn nhau giữa các ứng dụng và các thiết bị*". Trong thuật ngữ được sử dụng bởi PKCS#11, một thẻ (token) là một thiết bị lưu trữ các đối tượng và thực thi các tác vụ mật mã. Đây là đặc tính logic chứ không phải là đặc tính vật lý. Trong đó, một thiết bị có thể bao gồm một số thẻ logic riêng biệt. Khi muốn sử dụng một thẻ, trước hết người dùng phải thiết lập một phiên làm việc với thẻ, điều này yêu cầu người dùng phải đăng nhập và được xác thực đối với thiết bị. Sau đó, người dùng có thể sử dụng chức năng được cung cấp bởi thẻ, bằng cách thực hiện các lời gọi hàm thông qua giao diện hay giao diện lập trình ứng dụng (Application Programming Interface - API). Các đối tượng được đặc trưng hóa thành các đối tượng thẻ (token objects) hoặc các đối tượng phiên (session objects).

Trong lịch sử phát triển, hầu hết các khía cạnh cần thiết đối với một chuẩn giao tiếp giữa các nhà phát triển thiết bị và những người triển khai ứng dụng đã được nghiên cứu đầy đủ, nên hầu hết các hãng nổi tiếng trên thế giới đã chấp nhận PKCS#11 như một chuẩn giao diện lập trình ứng dụng cho các sản phẩm bảo mật và an toàn thông tin. Ví dụ: các hãng nCipher, IBM, Thales, Rainbow, Utimaco, Safenet, AEP.... Tuy nhiên, cũng xuất phát từ sự đầy đủ, tính đa dạng mà chuẩn hỗ trợ đã dẫn đến một số vấn đề "*không rõ ràng*" gây nên những "*bẫy phức tạp*" ảnh hưởng đến độ an toàn khi triển khai cài đặt, sử dụng trong thực tế.

Việc phân tích, đánh giá độ an toàn của PKCS#11, cũng như các cài đặt cụ thể của nó đã được đề cập trong các tài liệu [2, 3]. Trong đó, tài liệu [2] đã đưa ra một số điểm yếu của PKCS#11 mà kẻ tấn công có thể khai thác, nhằm nhận được các thông tin mà theo mục tiêu thiết kế của PKCS#11 là được bảo vệ trên các thẻ. Các tài liệu được công bố sau đó hầu hết tập trung vào việc mô hình hóa và triển khai cài đặt các công cụ

nhằm đánh giá các sản phẩm hiện có trên thị trường đối với các điểm yếu nêu trên.

Bố cục của bài báo gồm 4 Mục. Sau Mục Giới thiệu, Mục II trình bày về một số điểm yếu của tiêu chuẩn PKCS#11; Mục III đưa ra những giải pháp nhằm khắc phục các điểm yếu khi thực hiện xây dựng, phát triển các môđun phần cứng mật mã (Hardware Secutiry Modulo -HSM) sử dụng PKCS#11 như một giao diện lập trình ứng dụng; Cuối cùng Mục IV là một số kết luận.

II. MỘT SỐ ĐIỂM YẾU CỦA PKCS#11

A. Cơ chế kiểm soát truy nhập

Trong tiêu chuẩn PKCS#11 đưa ra hai kiểu người dùng: các nhân viên an ninh (Security Officers - SO) và người dùng thông thường. Nhân viên an ninh chịu trách nhiệm quản trị người dùng thông thường, với các chức năng như: khởi tạo ban đầu, xóa các thẻ, thay đổi mật khẩu, hiển thị thông tin chung của thẻ. Khác với những người dùng thông thường, các nhân viên an ninh không có quyền thực hiện các tác vụ mật mã. Mọi người dùng đều phải được xác thực trước khi truy cập các đối tượng và sử dụng các chức năng của thẻ. Điều này đạt được thông qua việc sử dụng cơ chế xác thực với số định danh cá nhân (PIN), bản chất là cơ chế xác thực dựa trên mật khẩu. Cơ chế kiểm soát truy nhập theo chuẩn PKCS#11 có thể dẫn đến các điểm yếu tiềm năng dưới đây:

- Trong cơ chế xác thực đăng nhập, sử dụng tên người dùng và mật khẩu gặp những điểm yếu tiềm năng đối với một số tấn công tìm kiếm mật khẩu (cần chú ý trong ngữ cảnh chung của chuẩn, người dùng có thể thử tất cả các mật khẩu). Đặc biệt, ở đây chúng ta quan tâm nhiều hơn đến cơ chế kiểm soát đăng nhập đối với SO. Trong các hệ thống thông tin quan trọng sử dụng các thiết bị phần cứng mật mã (ví dụ như đối các hệ thống PKI của các quốc gia trong đó có sử dụng các thiết bị HSM phục vụ việc lưu trữ và thực thi các tác vụ mật mã an toàn), nếu chỉ một SO có toàn quyền quản trị, có thể xảy ra tình trạng lạm dụng vị trí, thẩm quyền của một cá nhân, từ đó làm tổn thương đến độ an toàn của toàn bộ hệ thống.
- Trong cơ chế kiểm soát truy nhập nói riêng và trong các phiên làm việc với các thiết bị nói chung, thông tin trao đổi khi sử dụng các hàm giao tiếp của PKCS#11 không được bảo vệ dễ dẫn đến tình trạng kẻ tấn công có thể nghe trộm, chèn, sửa đổi nội dung thông tin trao đổi, nhằm nhận được các thông tin nhạy cảm (khóa, nguyên liệu khóa,…) được lưu trên các thiết bị.

B. Một số điểm yếu đối với API khóa đối xứng

PKCS#11 hỗ trợ các hàm quản lý khóa mật mã dưới đây:

- *C_GenerateKey*: sinh ra một khóa bí mật;
- *C_GenerateKeyPair*: sinh ra cặp khóa công khai/riêng;
- *C_WrapKey*: bọc một khóa riêng hoặc khóa bí mật;
- *C_UnwrapKey*: mở bọc một khóa đã được bọc;
- *C_DeriveKey*: dẫn xuất ra một khóa từ một khóa cơ sở;

Ví dụ đối với hàm *C_WrapKey* có thể được dùng trong các tình huống sau:

- Bọc khóa bí mật bất kỳ bằng khóa công khai RSA.
- Bọc khóa bí mật bất kỳ bằng một khóa bí mật khác.
- Bọc khóa riêng RSA, Diffie-Hellman hoặc DSA bằng khóa bí mật.

Dưới đây là các điểm yếu đối với API khóa đối xứng, bao gồm:

Điểm yếu thứ nhất - khả năng "phù phép" khóa: "phù phép" khóa là kỹ thuật sinh bất hợp pháp các khóa trong thiết bị. Bond [4] đã nhận diện "phù phép" khóa như một hiểm họa an toàn đối với các thiết bị phần cứng mật mã. Bond đã nhận thấy rằng, nếu các khóa bí mật được lưu ở ngoài thiết bị chống can thiệp sẽ dẫn tới việc sinh khóa không hợp pháp. Ví dụ, đối với khóa DES, kẻ tấn công có thể sinh 64 bit ngẫu nhiên và dùng hàm *C_UnWrapKey* để nạp vào thiết bị, nếu việc kiểm tra tính chẵn lẻ của khóa là bắt buộc, thì có 1 trong 2^8 cơ hội "khóa" mới này sẽ có tính chẵn lẻ hợp lệ. Như vậy, các "khóa" mới có thể sẽ được sử dụng cho các chức năng như mã hóa dữ liệu, hay bọc các khóa khác khi xuất ra ngoài thiết bị.

Điểm yếu thứ hai - liên kết khóa: Hiểm họa này ban đầu cũng được đưa ra trong [4]. Việc thiếu liên kết mật mã giữa các nửa của khóa sử dụng trong thuật toán 3DES theo chế độ EDE có thể dẫn đến sự suy giảm về độ an toàn. Ý tưởng của việc khai thác điểm yếu này có thể thấy qua hai kịch bản tấn công dưới đây:

Trong [5] có đưa ra tấn công chung đối với thiết bị mật mã sử dụng 3DES theo chế độ EDE với hai khóa (k_1, k_2) với phép mã hóa dữ liệu *data* được thực hiện như sau: $E_{k_1}(D_{k_2}(E_{k_{12}}(data)))$. Nếu k_1, k_2 không có sự liên kết mật mã, đặc biệt là

trong trường hợp không có sự kiểm tra việc sử dụng lặp đi lặp lại các "nửa" khóa, bằng cách nào đó kẻ tấn công có thể sửa đổi khóa 3DES thành (k_1, k_1) hoặc (k_2, k_2) thì dữ liệu *data* thay vì được mã hóa bởi thuật toán 3DES với 112 bit khóa thì sẽ được mã hóa bằng thuật toán DES đơn với 56 bit khóa.

Trong [1] cũng giới thiệu tấn công đối với các thiết bị mật mã sử dụng 3DES như sau:

- Sử dụng hàm *C_WrapKey* để xuất khóa 3DES là cặp có thứ tự (k_1, k_2), chú ý rằng mỗi nửa được mã hóa độc lập $\langle E_{KEK}(k_1), E_{KEK}(k_2)\rangle$
- Sử dụng hàm *C_UnWrapKey* để nạp lại nửa khóa thứ nhất đã được xuất như một khóa có độ dài đơn được mã hóa ở chế độ ECB: $D_{KEK}(E_{KEK}(k_1)) = k_1$.
- Sử dụng hàm *C_UnWrapKey* để nạp lại nửa khóa thứ hai đã được xuất như một khóa có độ dài đơn được mã hóa ở chế độ ECB (sử dụng cùng *KEK*); $D_{KEK}(E_{KEK}(k_2)) = k_2$.
- Thực hiện tấn công tìm kiếm khóa với mỗi khóa độ dài đơn (k_1, k_2) một cách riêng biệt.

Điểm yếu thứ ba - sự tách biệt khóa: Các đối tượng khóa bí mật trong PKCS#11 cho phép chỉ ra mục đích khóa được sử dụng cho các tác vụ bao gồm: phép mã hóa, giải mã, sinh MAC, kiểm tra MAC, bọc khóa và mở bọc khóa. Cụ thể thông qua các thuộc tính trong Bảng 1 dưới đây:

BẢNG 1. CÁC THUỘC TÍNH KIỂM TRA

Thuộc tính	Giá trị	Ý nghĩa
CKA_ENCRYPT	CK_BBOOL	TRUE nếu hỗ trợ mã hóa
CKA_DECRYPT	CK_BBOOL	TRUE nếu hỗ trợ giải mã
CKA_SIGN	CK_BBOOL	TRUE nếu hỗ trợ các ký
CKA_VERIFY	CK_BBOOL	TRUE nếu hỗ trợ kiểm tra
CKA_WRAP	CK_BBOOL	TRUE nếu hỗ trợ bọc
CKA_UNWRAP	CK_BBOOL	TRUE nếu hỗ trợ mở bọc

Tuy nhiên, do các thuộc tính trên có thể được đặc tả một cách độc lập, điều này dẫn đến rủi ro mất an toàn được khai thác dưới dạng một tấn công tách biệt khóa. Một ví dụ cụ thể, giả sử khóa *K* được thiết lập các thuộc tính *CKA_WRAP* và *CKA_DECRYPT* là TRUE, kẻ tấn công có thể thực hiện:

- Xuất khóa mục tiêu (K_{target}) với khóa *KEK* là *K* sử dụng hàm *C_WrapKey*, nhận được kết quả $T = E_K(K_{target})$.
- Giải mã thẻ kết quả sử dụng hàm *C_Decrypt* với khóa *K* như một khóa giải mã dữ liệu. Kết quả nhận được $D_K(T) = D_K(E_K(K_{target}))= K_{target}$.

Trong chuẩn PKCS#11, thuộc tính của một đối tượng khóa bí mật trong cài đặt cụ thể có thể sửa đổi thông qua việc gọi hàm *C_SetAttributeValue* hoặc trong tiến trình sao chép đối tượng sử dụng hàm *C_CopyObject*, nên kẻ tấn công có thể khai thác để thiết lập các thuộc tính nêu trên của các đối tượng khóa đã tồn tại.

Điểm yếu thứ tư - các khóa hoặc thuật toán yếu hơn: Trong chuẩn PKCS#11 hỗ trợ nhiều thuật toán với độ dài khóa khác nhau như RC2, DES, IDEA, AES. Hơn nữa, trong đặc tả PKCS#11 cho phép bọc một khóa bởi một khóa thứ hai có độ dài ngắn hơn. Điều này sẽ dẫn tới rủi ro mất an toàn có thể bị khai thác bởi kẻ tấn công như sau: giả sử kẻ tấn công đặt mục tiêu khôi phục khóa 3DES có độ dài gấp đôi $k = (k_1, k_2)$, anh ta thực hiện như sau:

- Xuất khóa *k* được mã hóa bởi khóa độ dài đơn (*KEK*), nhận được $T = E_{KEK}(k)$
- Xuất khóa độ dài đơn (*KEK*) được mã hóa bởi bản thân nó cho kết quả $T_{KEK} = E_{KEK}(KEK)$, có thể sử dụng thuật toán và chế độ yếu hơn như CKC_DES_ECB.
- Tấn công khôi phục khóa độ dài đơn KEK.
- Khi khóa độ dài đơn KEK bị khôi phục, kẻ tấn công có thể khôi phục khóa có độ dài gấp đôi ban đầu.

Điểm yếu thứ năm - không gian khóa suy giảm: Khi một đối tượng khóa không được thiết lập cờ "unextractable", kẻ tấn công có thể sử dụng cơ chế CKM_EXTRACT_KEY_FROM_KEY để trích một phần các bit từ khóa cho trước để tạo khóa ngắn hơn. Từ đó tác động làm giảm không gian khóa trong quá trình tìm kiếm khóa mục tiêu. Ví dụ, người ta có thể trích 40 bit từ khóa dùng cho DES để tạo 40 bit khóa dùng cho RC2, sau đó có thể thực hiện tìm kiếm bởi các công cụ vét cạn. Không gian khóa thực sự có thể nhỏ hơn, bởi có các bit kiểm tra trong khóa DES. Còn lại 24 bit (ít hơn 3 bit kiểm tra) của khóa DES ban đầu có thể được tìm kiếm độc lập.

Điểm yếu thứ sáu - tìm kiếm song song: CKM_XOR_BASE_AND_DATA cung cấp một phương pháp để tạo một khóa mới trên cơ sở thực

hiện phép XOR các mẫu đã biết với một khóa đã có. Điều này có thể được khai thác bởi kẻ tấn công, nhằm giảm không gian khóa và tìm kiếm khóa mục tiêu thông qua việc sinh số lượng lớn các khóa quan hệ. Một ví dụ cụ thể, kẻ tấn công cần khám phá khóa mục tiêu K_{target}, anh ta thực hiện như sau:

- Sinh tập 2^{16} các khóa quan hệ đã biết của khóa mục tiêu ban đầu $\{K_i | K_i = K_{target} \oplus \Delta_i, i = 1, \dots, 2^{16}\}$ với $\Delta_i \neq \Delta_j$ với $i, j \leq 2^{16}$ và Δ_i là giá trị khác 0.
- Sử dụng mỗi khóa, mã hóa một mẫu đã biết (P) và lưu kết quả trong cơ sở dữ liệu có thể tìm kiếm $\{C_i | C_i = e_{K_i}(P), i = 1, \dots, 2^{16}\}$.
- Tìm khóa bằng cách lặp các phép mã hóa của mẫu đã biết (P) và so sánh kết quả với các chỉ mục trong cơ sở dữ liệu.
- Sau khi thử trung bình 2^{39} phép mã hóa, chúng ta mong đợi sẽ tìm thấy sự tương thích (tìm được khóa K_i mà cho đầu ra được mã hóa trong cơ sở dữ liệu).
- Khôi phục khóa mục tiêu ban đầu $K_{target} = K_i \oplus \Delta_i$

Điểm yếu thứ bảy - Tấn công khóa quan hệ: Cũng như ở điểm yếu thứ sáu, thông qua việc sử dụng cơ chế CKM_XOR_BASE_AND_DATA, kẻ tấn công có thể tạo một tập các khóa quan hệ để thực hiện tấn công khóa quan hệ. Điều này có thể được sử dụng để giảm 3-key DES xuống chỉ mạnh hơn không đáng kể so với DES (giảm tìm kiếm không gian khóa tới 2^{56} phép toán để cô lập một thành phần khóa). Tấn công có thể được mô tả như sau: sử dụng cặp khóa quan hệ $K_1 =< k_1, k_2, k_3 >$, $K_2 =< k_1 \oplus \Delta, k_2, k_3 >$, mã hóa bản rõ P với khóa K_1, và giải mã bản mã với khóa K_2 cho kết quả P'. Thì $C = E_{K_1}(P)$, $P' = D_{K_2}(C)$ bởi vậy $P' = D_{K_2}\left(E_{K_1}(P)\right)$. Sử dụng 3DES ở chế độ EDE sẽ nhận được:

$$P' = D_{k_1 \oplus \Delta}\left(E_{k_2}\left(D_{k_3}\left(E_{k_3}\left(D_{k_2}\left(E_{k_1}(P)\right)\right)\right)\right)\right) = d_{k_1 \oplus \Delta}\left(E_{k_1}(P)\right)$$

Bởi vậy, k_1 đã bị cô lập thành công và có thể được khôi phục độc lập với k_2 và k_3, cụ thể thông qua tìm kiếm khóa vét cạn. Công việc yêu cầu tìm kiếm trung bình 2^{56} phép toán DES đơn.

C. Một số điểm yếu đối với API khóa công khai

Mục này tập trung vào việc xem xét các tấn công liên quan đến việc sử dụng các chức năng API khóa công khai. Chúng ta bắt đầu bằng việc xem lại các chức năng hàm *C_WrapKey* và xem xét việc bọc các khóa riêng RSA bằng các khóa đối xứng.

Trong chuẩn PKCS#11, một khóa riêng chỉ được xuất (và nhập) không chỉ chứa thành phần bí mật và môđun mà còn có cả thành phần công khai và thông tin CRT. Thông tin này có định dạng BER theo kiểu RSAPrivateKey ASN.1 được quy định trong PKCS#11. Chuỗi các byte kết quả được mã hóa với khóa bí mật ở chế độ CBC và với phần đệm tuân theo PKCS.

BẢNG 2. KIỂU DỮ LIỆU VÀ Ý NGHĨA CÁC THUỘC TÍNH

Thuộc tính	Kiểu dữ liệu	Ý nghĩa
CKA_MODULUS	Số lớn	Modulo *n*
CKA_PUBLIC_EXPONENT	Số lớn	Thành phần công khai *e*
CKA_PRIVATE_EXPONENT	Số lớn	Thành phần bí mật *d*
CKA_PRIME_1	Số lớn	Số nguyên tố *p*
CKA_PRIME_2	Số lớn	Số nguyên tố *q*
CKA_EXPONENT_1	Số lớn	*d* mod *p*-1
CKA_EXPONENT_2	Số lớn	*d* mod *q*-1
CKA_COEFFICIENT	Số lớn	Hệ số CRT *q*-1 mod *p*

Bản mã CBC được giải mã, và đệm PKCS được loại bỏ. Dữ liệu nhận được sẽ được bóc tách ra kiểu PrivateKeyInfo, và sẽ cho ra khóa được bọc. Lỗi xảy ra nếu khóa được bọc ban đầu không được giải mã đúng, hoặc dữ liệu giải mã sau khi bỏ đệm không được phân tách chính xác, hoặc kiểu của nó không khớp với kiểu khóa được xác định trong mẫu đối với khóa mới. Cơ chế mở bọc chỉ đóng góp các thuộc tính được xác định trong kiểu PrivateKeyInfo đối với khóa được mở bọc mới, các thuộc tính khác phải được chỉ ra trong mẫu, hoặc sẽ lấy giá trị mặc định của chúng.

Điểm yếu thứ tám - sử dụng thuật toán hoặc khóa yếu hơn: Việc lựa chọn thuật toán khóa đối xứng (và chiều dài khóa) được sử dụng để bảo vệ khóa riêng RSA có thể dẫn đến các tấn công giống như trong điểm yếu thứ tư đã trình bày ở trên.

Điểm yếu thứ chín - định dạng khóa riêng: Để tăng tốc trong tính toán chữ ký số thông qua việc sử dụng định lý phần dư Trung Hoa (CRT), nhiều cài đặt PKCS#11 hỗ trợ kiểu lưu khóa riêng gồm tám thành phần *n*, *p*, *q*, *e*, *d*, *d* mod *p*-1, *d* mod *q*-1

và q-1 mod p. Tuy nhiên, đối với việc sử dụng định dạng khóa gồm 8 thành phần nêu trên các nguyên thủy mật mã có thể bị khá nhiều tấn công, ví dụ như các tấn công đã được công bố trong [5,6].

Điểm yếu thứ mười - sửa đổi khóa riêng: Ta cần xem xét sự tác động của việc thay thế một khối của bản mã (ví dụ khối khóa đã được bọc) với một giá trị khác. Khi khóa được mở bọc, sẽ gây ra một khối tương ứng của bản rõ cũng như các khối tiếp theo có các giá trị khác. Phần còn lại của khóa vẫn còn nguyên vẹn. Độ dài của các kiểu dữ liệu số lớn được ghi dạng BER phụ thuộc vào kích thước của các số lớn (thường là các số 512, 1024 hoặc 2048 bit). Dù ở trường hợp nào, chúng cũng bao gồm ít nhất một số khối. Bởi vậy kẻ tấn công có thể sửa đổi một trong các số lớn một cách độc lập đối với dữ liệu khác trong khóa riêng được bọc (bao gồm dữ liệu đệm ở cuối). Điều này có thể được sử dụng để thực hiện các tấn công phân tích lỗi như trong [7].

Điểm yếu thứ mười một - thành phần công khai nhỏ và không đệm: Do có lợi thế về tốc độ khi có số mũ công khai nhỏ với trọng số Hamming thấp, trong nhiều cài đặt API các số mũ công khai được chọn là 3 và 65537. Bên cạnh đó, PKCS#11 cũng hỗ trợ cơ chế bọc khóa bí mật dùng khóa công khai và không sử dụng kỹ thuật đệm. Số mũ công khai nhỏ, kỹ thuật mã hóa không đệm, hoặc đồng thời cả là điều kiện thuận lợi cho khá nhiều tấn công đã được công bố đối với hệ thống mật mã RSA. Một ví dụ minh họa đơn giản: khi bọc khóa bí mật k chúng ta có $T=k^e$ mod n. Nếu $k^e < n$ (tức là $e < \frac{log_2(n)}{log_2(k)} \leq \frac{log_2(n)}{128}$), thì $T = k^e$. Do đó k có thể được khôi phục là $k = T^{\frac{1}{e}}$.

Điểm yếu thứ mười hai - khóa công khai "Trojan": Như đã đề cập ở trên, các khóa công khai trong PKCS#11 API là các thẻ rõ không có các phép kiểm tra xác thực bổ sung. Do đó, có thể sử dụng bất kỳ khóa công khai nào làm đầu vào của hàm *C_WrapKey*. Điều này cho phép kẻ tấn công sử dụng một khóa công khai "Trojan" với khóa riêng tương ứng đã biết (thường kẻ tấn công sẽ sinh xác suất ra cặp khóa). Kẻ tấn công yêu cầu thẻ PKCS#11 xuất một khóa mục tiêu k dưới khóa công khai mà anh ta cung cấp, nhận được phúc đáp $T = k^e$ mod n. Vì anh ta có khóa riêng d tương ứng, nên anh ta dễ dàng khôi phục k. Phương pháp đơn giản này có thể được sử dụng để khôi phục tất cả các khóa có thể xuất kể cả chúng là khóa đối xứng hay khóa riêng.

Điểm yếu thứ mười ba - khóa được bọc "Trojan": Tương tự như sử dụng các khóa công khai không được chứng thực, điểm yếu này chính là không có phương pháp để kiểm tra rằng thẻ khóa được bọc thực sự là xác thực. Do đó, cho trước một thiết bị hỗ trợ PKCS#11 chứa khóa riêng (d, n), và kiến thức về giá trị của khóa công khai (e, n), kẻ tấn công chọn một khóa k tùy ý, sau đó bọc dưới khóa công khai đã biết để có $T = k^e$ *mod* n. Sau đó kẻ tấn công sẽ gọi hàm *C_UnWrapKey* cung cấp khóa được bọc Trojan T và tham chiếu đến khóa riêng bên trong thiết bị. Thẻ PKCS#11 tính T^d *mod* $n = k$ và nhập khóa đã biết k như một khóa mới vào trong hệ thống. Sau đó k được sử dụng để xuất các khóa khác từ thiết bị, mà anh ta có thể giải mã và khôi phục. Như vậy, tồn tại một yêu cầu nhằm cung cấp công cụ cho việc kiểm tra tính xác thực và nguồn gốc của khóa được bọc.

Ngoài ra, nếu khóa đối xứng được bọc bởi khóa công khai không chứa các thành phần tách biệt cũng gặp phải điểm yếu như điểm yếu thứ ba.

III. PHÂN TÍCH SỰ ẢNH HƯỞNG VÀ ĐỀ XUẤT GIẢI PHÁP KHẮC PHỤC

Mười ba điểm yếu đã được liệt kê ở trên hầu hết đều xuất phát từ việc kẻ tấn công có quyền truy nhập vào thiết bị, hoặc can thiệp vào thông tin trao đổi giữa ứng dụng và thiết bị trong quá trình thực hiện một phiên. Do đó, cần xem xét khả năng tăng cường cơ chế kiểm soát truy nhập và bảo vệ phiên làm việc giữa ứng dụng và thiết bị.

A. Đối với cơ chế kiểm soát truy nhập

Chuẩn PKCS#11 cho phép các nhà phát triển tăng cường thêm các kỹ thuật kiểm soát truy nhập hoặc thay thế kỹ thuật kiểm soát truy nhập người dùng mà chuẩn khuyến cáo bởi các kỹ thuật tùy biến khác của khách hàng. Dưới đây, chúng tôi đề xuất một số giải pháp nhằm tăng cường cho các kỹ thuật kiểm soát truy nhập người dùng khi xây dựng phát triển API dựa trên PKCS#11.

Đối với việc kiểm soát truy nhập SO: theo chuẩn PKCS#11 một SO không có quyền thực hiện các tác vụ mật mã trên thẻ, nhưng lại được thêm mới người dùng và có thể thay đổi mật khẩu của người dùng hợp pháp đã tồn tại. Như vậy một SO không trung thực có thể thay mật khẩu của một người dùng bởi một giá trị đã biết, nhằm thực hiện các tác vụ mật mã với vai trò là người dùng đó. Để tránh trường hợp rủi ro khi đặt sự an toàn

của cả một hệ thống vào một đối tượng quản trị duy nhất (đặc biệt là đối với các hệ thống thông tin lớn sử dụng các HSM) chúng ta có thể thay thế cơ chế kiểm soát truy cập của SO sử dụng mật khẩu bởi một lược đồ ngưỡng, chẳng hạn như lược đồ chia sẻ bí mật của Shamir [8]. Với việc lựa chọn các tham số (*k*, *n*) hợp lý cho lược đồ chia sẻ bí mật của Shamir chúng ta có thể đưa ra ngưỡng *k* đủ lớn để các nhân viên an ninh có thể hợp tác và đồng thuận nhằm đạt được thẩm quyền truy nhập và thực thi tác vụ. Đồng thời cũng có thể đưa ra ngưỡng thiểu số để các nhân viên an ninh có thể phủ quyết mục đích trên.

Đối với việc kiểm soát truy nhập người dùng thông thường: trên cơ sở khuyến cáo của PKCS#11, chúng ta có hai giải pháp nhằm nâng cao độ an toàn trong kiểm soát truy nhập đối với người dùng thông thường: Thứ nhất, thay thế hoàn toàn cơ chế kiểm soát truy nhập sử dụng mật khẩu bằng cơ chế kiểm soát truy nhập mạnh hơn như cơ chế sử dụng mật mã khóa công khai, sử dụng các hàm kiểm tra,…; Thứ hai, gia cố cơ chế kiểm soát truy nhập chuẩn đã khuyến cáo, đồng thời kết hợp với lược đồ mã hóa dựa trên mật khẩu, nhằm bảo vệ khóa bí mật, khóa riêng lưu trên thiết bị, như một số sản phẩm thương mại hiện đang áp dụng. Trong bài viết này, chúng tôi giới thiệu chủ yếu về giải pháp thứ hai, cụ thể như sau:

- Để tăng cường cơ chế kiểm soát dựa trên mật khẩu, có thể áp dụng một số giải pháp kỹ thuật như: đặt ra yêu cầu về độ mạnh của mật khẩu; khi số lần thử đăng nhập thất bại đạt ngưỡng cho trước, thiết bị sẽ "nghỉ" trong một khoảng thời gian, trước khi cho phép tiến trình đăng nhập lại (không áp dụng việc khóa thiết bị khi số lần đăng nhập thất bại đạt ngưỡng vì như vậy dễ dẫn đến các tấn công từ chối dịch vụ DDoS)....
- Để kết hợp giữa cơ chế kiểm soát truy nhập với cơ chế bảo vệ các thông tin nhạy cảm lưu trên thiết bị (khóa, nguyên liệu khóa,...) chúng ta có thể sử dụng các lược đồ mã hóa dựa trên mật khẩu [9]. Đối với việc sử dụng lược đồ mã hóa dựa trên mật khẩu ngoài nguyên thủy mã khối, chúng ta cần có sự lựa chọn hợp lý các tham số (hàm PRF, độ dài mật khẩu, số lần lặp *c* và độ dài *salt*) cho hàm dẫn xuất khóa dựa trên mật khẩu để lược đồ đạt được độ an toàn mong muốn.

Đối với việc bảo vệ thông tin trao đổi trong một phiên: để đảm bảo thông tin trao đổi trong một phiên được bảo mật và xác thực có thể thiết lập việc sử dụng các giao thức SSL/TLS, IKE/IPSec,… giữa ứng dụng của người dùng và thiết bị.

B. Đối với các API khóa mật mã

Đối với điểm yếu thứ nhất: "phù phép" khóa không dẫn đến việc trực tiếp tiết lộ thông tin về "khóa" được dùng cho việc bọc khóa khác cũng như trong mã hóa dữ liệu. Tuy nhiên, với việc "phù phép" thành công các khóa đã chỉ ra sự không chặt chẽ trong quá trình quản lý và sử dụng khóa thông qua các API. Nghiêm trọng hơn, thông qua "phù phép" khóa, kẻ tấn công có thể thay thế thành công các khóa bí mật đã được kiểm tra các tiêu chuẩn về mật mã bởi các "khóa" ngẫu nhiên khác. Để khắc phục điểm yếu này, có thể áp dụng các cơ chế xác thực thông tin khóa rõ trước khi đưa vào sử dụng, thông qua việc sử dụng MAC hoặc chữ ký số.

Đối với điểm yếu thứ hai: qua hai kịch bản tấn công được đưa ra trong [2, 4], chúng ta thấy điểm yếu thứ hai chỉ áp dụng với các nguyên thủy mật khóa đối xứng sử dụng khóa gồm hai hoặc nhiều thành phần khác nhau không có sự "gắn kết". Để khắc phục điểm yếu này, giải pháp chung là cần áp dụng cơ chế đảm bảo tính "gắn kết" giữa các thành phần của khóa sau khi đã bọc. Ví dụ như sử dụng MAC hoặc chữ ký số như trong điểm yếu thứ nhất. Riêng đối với các thiết bị phần cứng mật mã chuyên dụng sử dụng trong các hệ thống thông tin cần độ an toàn cao có thể bỏ qua sự hỗ trợ của các API đối với 3DES, bằng cách loại bỏ các cơ chế CKM_DES3_ECB, CKM_DES3_CBC, …

Đối với điểm yếu thứ ba: đây là một điểm yếu khá nghiêm trọng, trong quá trình xây dựng và sử dụng các API có thể dẫn đến việc bị lộ khóa bí mật. Giải pháp chung và tuân thủ chuẩn PKCS#11 nhằm khắc phục điểm yếu này là không cho phép sửa đổi thuộc tính của một đối tượng khóa bí mật thông qua việc gọi hàm *C_SetAttributeValue* hoặc *C_CopyObject*.

Đối với điểm yếu thứ tư: điểm yếu này của PKCS#11 tương tự điểm yếu đã được đề cập đến trong tấn công "*Cipher Suite Rollback*" đối SSL/TLS. Phương pháp chung nhằm khắc phục điểm yếu này là chỉ hỗ trợ các thuật toán, độ dài khóa có độ an toàn tương đương, thông qua các định nghĩa dạng CKM_<NAME>_<MODE>. Đối với các thiết bị phần cứng mật mã chuyên dụng thậm chí chỉ nên dùng một thuật toán mã hóa khóa đối xứng duy nhất.

Đối với điểm yếu thứ năm: ngay trong giải pháp khắc phục điểm yếu thứ tư đã bao gồm một phần giải pháp khắc phục điểm yếu thứ năm. Tuy

nhiên, để tránh trường hợp một khóa được sinh bởi một khóa khác đã tồn tại, chúng tôi đề xuất giải pháp không sử dụng các cơ chế dẫn xuất khóa được định nghĩa bởi:

- CKM_CONCATENATE_BASE_AND_KEY, dẫn xuất một khóa bí mật từ kết quả nối hai khóa bí mật đã có.
- CKM_CONCATENATE_BASE_AND_DATA, dẫn xuất một khóa bí mật từ kết quả nối dữ liệu vào khóa bí mật đã chỉ ra.
- CKM_CONCATENATE_BASE_AND_BASE, dẫn xuất một khóa bí mật từ kết quả thêm dữ liệu vào tiền tố khóa bí mật đã chỉ ra.
- CKM_XOR_BASE_AND_DATA, là kỹ thuật hỗ trợ khả năng dẫn xuất một khóa bí mật thông qua thực hiện phép XOR khóa được trỏ bởi handle khóa cơ sở và dữ liệu.
- CKM_EXTRACT_KEY_FROM_KEY, cung cấp khả năng tạo một khóa bí mật từ các bit của một khóa bí mật khác.

Đối với điểm yếu thứ sáu và thứ bảy: các điểm yếu này xuất phát từ việc hỗ trợ cơ chế CKM_XOR_BASE_AND_DATA nhằm cung cấp phương pháp để tạo một khóa trên cơ sở XOR mẫu đã biết với một khóa đã có, nên giải pháp khắc phục hiểm họa thứ năm đã bao hàm giải pháp khắc phục hiểm họa thứ sáu và thứ bảy.

Đối với điểm yếu thứ tám: điểm yếu này xuất phát từ việc lựa chọn các bộ thuật toán mật mã không cùng độ an toàn, khi đó có thể xảy ra trường hợp một khóa riêng sử dụng cho lược đồ chữ ký số có độ an toàn 2^{128} được bọc bởi thuật toán mật mã khóa đối xứng có độ an toàn 2^{64} và được xuất ra ngoài thiết bị. Để khắc phục điểm yếu này, khi cài đặt, phát triển giao tiếp PKCS#11 cần lựa chọn thuật toán bọc khóa riêng (trước khi xuất ra ngoài thiết bị) có độ an toàn tương đương với hệ thống mật mã khóa công khai sử dụng khóa riêng đó có thể cung cấp, chẳng hạn để bọc khóa riêng RSA-3072 bit chỉ hỗ trợ các cơ chế có độ an toàn tối thiểu CKM_AES_128. Về độ an toàn tương đương của các thuật toán mật mã, đã được trình bày trong NIST SP 800-57 [10].

Đối với điểm yếu thứ chín: mặc dù chuẩn PKCS#11 định nghĩa đối tượng khóa riêng RSA gồm 8 thuộc tính phục vụ cho việc lưu trữ khóa riêng gồm 8 thành phần (n, p, q, e, d, d mod p-1, d mod q-1 và q-1 mod p). Tuy nhiên, chuẩn cũng cho phép lựa chọn giải pháp lưu khóa riêng RSA chỉ gồm hai thành phần (n, d) sử dụng các thuộc tính CKA_MODULUS và CKA_PUBLIC_EXPONENT. Do vậy, chúng tôi đề xuất sử dụng giải pháp lưu khóa riêng RSA gồm hai thành phần (n, d) nhằm khắc phục hiểm họa thứ chín. Cần chú ý, trong cài đặt, ngoài việc loại các thuộc tính còn lại như CKA_PRIME_1, CKA_PRIME_2,… khỏi định nghĩa đối tượng khóa riêng RSA thì cũng cần loại bỏ cơ chế sinh khóa RSA cho đầu ra khóa riêng gồm 8 thành phần, cũng như việc tính toán trong các lược đồ giải mã/ký mà hỗ trợ kiểu tính toán sử dụng 8 thành phần của khóa riêng.

Đối với điểm yếu thứ mười: để khắc điểm yếu này có thể sử dụng các kỹ thuật nhằm đảm bảo tính toàn vẹn của khóa riêng như hàm băm mật mã, MAC. Tuy nhiên, cần lưu ý rằng, khóa riêng khi được lưu bao giờ cũng được mã hóa bởi thuật toán mật mã khóa bí mật với một chế độ định trước. Do vậy, chúng tôi đề xuất bổ sung chế độ vừa mã hóa vừa xác thực sử dụng cho việc bảo vệ khóa riêng trong chế độ vừa mã hóa vừa xác thực. Chú ý rằng, PKCS#11 chỉ hỗ trợ hai chế độ đối với mã khối là ECB và CBC.

Đối với điểm yếu thứ mười một: đây là điểm yếu chung đối với hệ thống mật mã RSA. Điểm yếu này có thể bị kẻ tấn công khai thác khi xảy ra một trong hai trường hợp: số mũ công khai phải nhỏ hoặc lược đồ mã hóa khóa công khai không có thao tác đệm (padding). Do vậy, giải pháp chung để khắc phục tấn công này đã được mô tả trong Mục II, khi khai thác hiểm họa thứ mười một là đảm bảo đồng thời cả hai điều kiện không xảy ra.

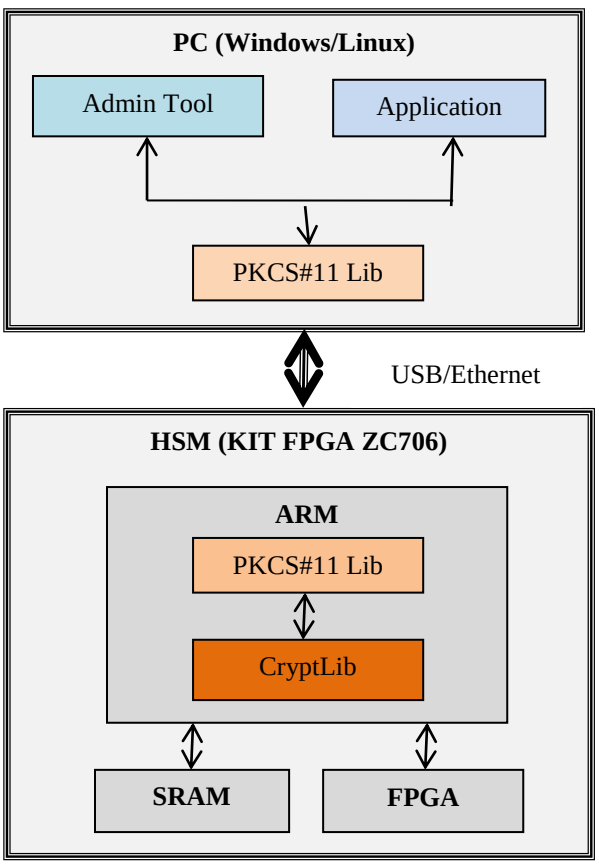
- Đối với điều kiện thứ nhất theo NIST 186-4 số mũ công khai phải lớn hơn 2^{16}. Tuy nhiên, chúng tôi đề xuất khi dùng thiết bị bảo mật phần cứng với các ứng dụng thương mại chúng ta có thể sử dụng các số mũ bí mật e cỡ 32 bit, còn đối với các thiết bị phần cứng chuyên dụng, chúng ta có thể sử dụng các số mũ công khai lớn hơn, chẳng hạn tối thiểu 160 bit. Cần chú ý rằng khi đưa điều kiện với độ dài tối thiểu của số mũ công khai e, cần phải xem xét trong mối quan hệ với độ lớn của số mũ bí mật d.
- Đối với điều kiện thứ hai, trong chuẩn PKCS#11 hỗ trợ hai cơ chế sử dụng khóa công khai để bọc các khóa bí mật sử dụng cho các nguyên thủy mật mã khóa đối xứng, gồm: CKM_RSA_PKCS và CKM_RSA_X_509. Trong đó, cơ chế CKM_RSA_X_509 không sử dụng kỹ thuật đệm, chỉ dùng RSA ở dạng nguyên thủy. Do vậy, chúng tôi đề xuất không sử dụng cơ chế CKM_RSA_X_509.

Đối với điểm yếu thứ mười hai: điểm yếu này xuất phát từ nguyên nhân trong chuẩn không yêu cầu kiểm tra một khóa công khai có thẩm quyền dùng để bọc khóa, trước khi được xuất ra ngoài thiết bị hay không. Do vậy, giải pháp để khắc phục điểm yếu này là khóa công khai cần được chứng thực trước khi dùng và trong quá trình dùng phải có thao tác kiểm tra về thẩm quyền bọc và xuất khóa ra ngoài thiết bị.

Đối với điểm yếu thứ mười ba: tương tự như đối với điểm yếu thứ mười hai, để khắc phục điểm yếu thứ mười ba cần áp dụng cơ chế xác thực đối với các khóa bí mật được bọc bởi các khóa công khai, cũng như chứng thực và kiểm tra thẩm quyền của khóa công khai khi sử dụng khóa riêng tương ứng để mở bọc (sử dụng hàm *C_UnWrapKey*) một khóa đối xứng.

IV. MỘT SỐ KẾT QUẢ THỰC HÀNH

Để minh họa cho việc triển khai một số giải pháp đã trình bày trong Mục III, chúng tôi xây dựng mô phỏng một thiết bị HSM sử dụng KIT FPGA ZC706.



Hình 1. Mô hình thiết bị HSM

Mô hình thiết bị HSM được mô tả trong Hình 1, bao gồm:

- **Admin Tool**: Công cụ quản trị (các chức năng của SO trong chuẩn PKCS#11).
- **Application**: Các ứng dụng do người dùng phát triển sử dụng các chức năng của HSM thông qua giao diện PKCS#11.
- **PKCS#11 Lib**: Thư viện hỗ trợ các hàm giao tiếp dựa trên chuẩn PKCS#11.
- **CryptLib**: Thư viện hỗ trợ một số thuật toán lược đồ như RSA-PSS, RSA-OAEP, ECDSA, AES, SHA-2, …
- **FPGA**: Thực hiện một số phép tính toán như: lũy thừa môđun, phép nhân điểm trên đường cong elliptic.
- **SRAM**: lưu khóa riêng, nguyên liệu khóa được sử dụng trong việc mã khóa riêng, các dữ liệu phục vụ việc xác thực.

B. Một số giải pháp được áp dụng

Các giải pháp đưa ra trong Mục III đều có thể được thực hiện trong quá cài đặt API cho thiết bị (liên quan đến 4 thành phần chính Admin Tool, Application, PKCS#11 Lib và CryptLib). Dưới đây chúng tôi minh họa sơ lược kết quả cài đặt thực hiện 3 giải pháp khắc phục đại diện cho 3 nhóm các điểm yếu đã đưa ra trong Mục II.

Thực hiện giải pháp kiểm soát truy nhập SO

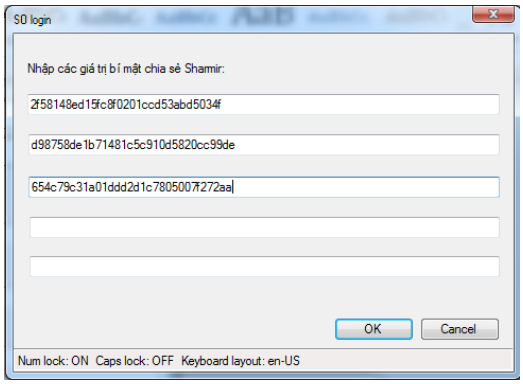
Để thực hiện giải pháp nâng cao độ an toàn trong kiểm soát truy nhập SO, chúng tôi sử dụng lược đồ chia sẻ bí mật của Shamir ([3, 5]) trên trường Z_p với p là số nguyên tố 128 bit.

- Khi khởi tạo token từ Admin Tool, HSM sinh giá trị bí mật và lưu trên SRAM; sinh các giá trị chia sẻ và gửi lại cho Admin Tool. Mỗi giá trị chia sẻ sẽ được cấp cho một SO (ví dụ trong Hình 2).

```
Khoi tao Token
Sinh cac gia tri chia se bi mat su dung luoc do Shamir (3,5)

Gia tri chia se bi mat thu 1:
2f58148ed15fc8f0201ccd53abd5034f

Gia tri chia se bi mat thu 2:
d98758de1b71481c5c910d5820cc99de

Gia tri chia se bi mat thu 3:
654c79c31a01ddd2d1c7805007f272aa

Gia tri chia se bi mat thu 4:
c9b352d7606f8d84e32586135dcf9583

Gia tri chia se bi mat thu 5:
757873ca611f184a6e730b1b7af17ee5
```

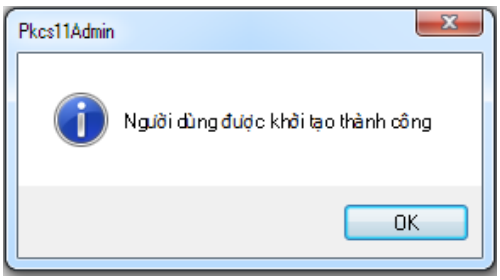
Hình 2. Ví dụ khởi tạo token

- Khi muốn đăng nhập HSM với vai trò SO để tạo người dùng (hoặc thực hiện một tác vụ nào đó khác), phải cung cấp ít nhất 3 giá trị chia sẻ từ 3 nhân viên an ninh. Giao diện nhập như trong Hình 3.



Hình 3. Bảng nhập các giá trị

- Từ các giá trị chia sẻ HSM sẽ tính giá trị bí mật, so sánh với giá trị bí mật đã lưu, nếu giống nhau quá trình đăng nhập và thực hiện tạo người dùng thành công sẽ được thông báo như Hình 4.



Hình 4. Thông báo khởi tạo thành công

Thực hiện giải pháp khắc phục điểm yếu thứ ba như sau:

Để khắc phục điểm yếu thứ ba, khi tạo đối tượng khóa bí mật, thuộc tính CKA_MODIFIABLE luôn mặc định được thiết lập giá trị false.

```
privateKeyAttributes.Add(new
ObjectAttribute(CKA.CKA_MODIFIABLE,
false));
```

Khi đó đối tượng khóa được tạo sẽ không thể thay đổi các thuộc tính.

Thực hiện giải pháp khắc phục điểm yếu thứ chín như sau:

Để khắc phục điểm yếu thứ chín trong cài đặt chúng tôi sử dụng khóa RSA với hai thành phần (n, d):

- Khi khai báo, hai tham số (n, d) được lưu trong vùng được bảo vệ của lớp RSAPrivateKey:

```
class RSAPrivateKey : public PrivateKey
{
Public:
…
Protected:
ByteString d, n;
}
```

- Trong thư viện mật mã CryptLib hàm sinh khóa RSA sẽ tạo tham số với hai thành phần (n, d).
- Sau khi (n, d) được tạo sẽ được lưu vào đối tượng khóa bí mật :

```
osobject->setAttribute(CKA_MODULUS,
modulus);
osobject-
>setAttribute(CKA_PRIVATE_EXPONENT,
privateExponent);
```

Thay vì lưu 8 thành phần như trong chuẩn PKCS#11 đã khuyến cáo như dưới đây:

```
osobject->setAttribute(CKA_MODULUS,
modulus);
osobject-
>setAttribute(CKA_PRIVATE_EXPONENT,
privateExponent);
osobject-
>setAttribute(CKA_PUBLIC_EXPONENT,
publicExponent);
osobject->setAttribute(CKA_PRIME_1,
prime1);
osobject->setAttribute(CKA_PRIME_2,
prime2);
osobject-
>setAttribute(CKA_EXPONENT_1,exponent1);
osobject->setAttribute(CKA_EXPONENT_2,
exponent2);
osobject->setAttribute(CKA_COEFFICIENT,
coefficient);
```

VI. KẾT LUẬN

Trên cơ sở tổng hợp và giới thiệu một số điểm yếu đối với PKCS#11, chúng tôi đã phân tích sự ảnh hưởng và đề xuất ra các giải pháp khắc phục đối với từng điểm yếu. Các giải pháp khắc phục là hoàn toàn có thể thực hiện được trong quá trình cài đặt PKCS#11 với vai trò là API của các thiết bị phần cứng mật mã. Tuy nhiên, để có được các thiết bị phần cứng mật mã (PKI Token, HSM,…)

thật sự an toàn, đặc biệt là các thiết bị phần cứng mật mã chuyên dụng, chúng ta cần xem xét và có những sửa đổi, bổ sung hợp lý đối với PKCS#11. Ví dụ, hiện tại PKCS#11 chỉ hỗ trợ hai chế độ ECB và CBC đối với mã khối; ở chế độ CBC sử dụng phương pháp đệm PKCS#5 là không an toàn; lược đồ chữ ký RSA hỗ trợ cả hai phiên bản PKCS#1 1.5 và 2.1.

TÀI LIỆU THAM KHẢO

[1] RSA Laboratories. "PKCS #11 v2.20: Cryptographic Token Interface Standard", RSA Security Inc., 2004

[2] Jolyon Clulow. "On the security of PKCS#11", Springer-Verlag Berlin Heidelberg, 2003.

[3] Matteo Bortolozzo, Matteo Centenaro, Riccardo Focardi, Graham Steel. "Attacking and Fixing PKCS#11 Security Tokens", Copyright 2010 ACM, 2010.

[4] Mike Bond. "Attacks on Cryptoprocessor Transaction Sets" Springer-Verlag Berlin Heidelberg 2001.

[5] Eric Brier, David Naccache, Phong Q. Nguyen, Mehdi Tibouchi. "Modulus Fault Attacks Against RSA-CRT Signatures".

https://eprint.iacr.org/2011/388.pdf.

[6] Abderrahmane Nitaj, "A new attack on RSA and CRT-RSA", AFRICACRYPT 2012.

[7] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton, "On the importance of checking cryptographic protocols for faults". In Advances in Cryptology EUROCRYPT '97, vol. 1233, pp. 37-51, 1997.

[8] Adi Shamir, "How to share a secret". 1979.

[9] RSA Laboratories. PKCS #5 v2.1: "Password-Based Cryptography Standard". RSA Security Inc., 2012.

[10] NIST SP 800-57 , "Recommendation for Key management - Part 1: General (revised)",2007.

SƠ LƯỢC VỀ TÁC GIẢ

TS. Hoàng Văn Thức

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

Email: thuchv@yahoo.com

Nhận bằng kỹ sư năm 1998 và Thạc sĩ năm 2004 chuyên ngành Kỹ thuật mật mã, Học viện Kỹ thuật mật mã. Nhận bằng Tiến sĩ Toán học, Viện Khoa học - Công nghệ quân sự năm 2012.

Hướng nghiên cứu hiện nay: Khoa học - Công nghệ Mật mã.

KS. Trần Sỹ Nam

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

Email: transynam1989@gmail.com

Nhận bằng kỹ sư chuyên ngành An toàn thông tin hệ thống viễn thông, Học viện FSO, Liên bang Nga năm 2013.

Hướng nghiên cứu hiện nay: Công nghệ mạng và bảo mật mạng.