**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**

**PTN CÔNG NGHỆ PHẦN MỀM**

- - - - 🙞🙞🟔🙜🙜 - - - -

**USB DONGLE**

**BÁO CÁO GIỮA KỲ**

**NĂM 2010**

# MỤC LỤC

MỤC LỤC 2

Chương 1 TỔNG QUAN 4

1.1 Mục đích 4

1.2 Nền tảng lý thuyết 4

1.2.1 Thuật toán AES 4

1.2.2 Thuật toán RSA 7

1.2.2.1 Tạo khóa 7

1.2.2.2 Mã hóa 8

1.2.2.3 Giải mã 8

1.3 Mô hình tổng thể 8

Chương 2 CHI TIẾT MÔ HÌNH 10

2.1 Xác nhận license 10

2.1.1 Điều kiện kích hoạt quá trình: 10

2.1.2 Các bước thực hiện 10

2.1.3 Các vấn đề 10

2.2 Xác nhận tồn tại thiết bị 11

2.2.1 Điều kiện kích hoạt quá trình 11

2.2.2 Các bước thực hiện 11

2.2.3 Các vấn đề 12

2.2.3.1 Tràn số trong bộ đếm 12

2.2.3.2 Hàm G 12

2.3 Giao tiếp giữa thiết bị và host 13

Chương 3 TIẾN ĐỘ THỰC HIỆN 14

3.1 Mô hình hiện tại 14

3.1.1 Cách thực hiện 14

3.1.2 Nhận xét 14

3.2 Công việc kế tiếp 14

# TỔNG QUAN

## Mục đích

Tài liệu được viết để mô tả cách tiếp cận và xây dựng mô hình cho ứng dụng USB Dongle. Trong đó, các vấn đề lý thuyết sẽ được trình bày một cách tóm tắt và mô hình sẽ được mô tả ở khía cạnh ứng dụng. Đối với các mô hình đưa ra, sẽ có phần đánh giá ưu khuyết điểm, các ý kiến chủ quan cũng như thực nghiệm khách quan và hướng giải quyết cho các nhược điểm của mô hình.

## Nền tảng lý thuyết

Phần cốt lõi của hệ thống chính là việc ứng dụng mật mã học vào quá trình chứng thực. Ở đây, ta đề cập đến hai phương pháp mã hóa khóa đối xứng và bất đối xứng. Ứng với từng phương pháp, thuật toán được chọn để sử dụng là AES (Advanced Encryption Standard) và RSA.

### Thuật toán AES

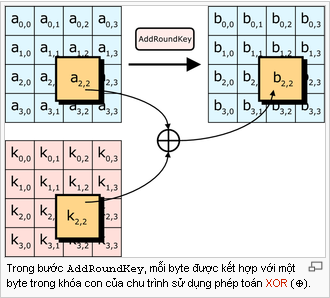
AES là thuật toán mã hóa khóa đối xứng, làm việc với từng khối dữ liệu 4×4 byte . Ở đây, ta chỉ xem xét AES với độ dài khóa là 128 bit. Quá trình mã hóa bao gồm 4 bước:

* AddRoundKey — mỗi byte của khối được kết hợp với khóa con, các khóa con này được tạo ra từ quá trình tạo khóa con Rijndael.
* SubBytes — đây là phép thế (phi tuyến) trong đó mỗi byte sẽ được thế bằng một byte khác theo bảng tra Rijndael S-box.
* ShiftRows — đổi chỗ, các hàng trong khối được dịch vòng.
* MixColumns — quá trình trộn làm việc theo các cột trong khối theo một phép biến đổi tuyến tính.

Tại chu trình cuối thì bước MixColumns được thay thế bằng bước AddRoundKey

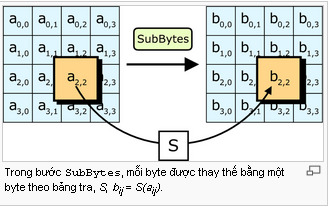
* Bước AddRoundKey

Tại bước này, khóa con được kết hợp với các khối. Khóa con trong mỗi chu trình được tạo ra từ khóa chính với quá trình tạo khóa con Rijndael; mỗi khóa con có độ dài giống như các khối. Quá trình kết hợp được thực hiện bằng cách XOR từng bít của khóa con với khối dữ liệu.



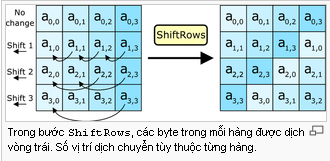
* Bước SubBytes

Các byte được thế thông qua bảng tra S-box. Đây chính là quá trình phi tuyến của thuật toán. Hộp S-box này được tạo ra từ một phép nghịch đảo trong trường hữu hạn GF (*28*) có tính chất phi tuyến. Để chống lại các tấn công dựa trên các đặc tính đại số, hộp S-box này được tạo nên bằng cách kết hợp phép nghịch đảo với một phép biến đổi affine khả nghịch. Hộp S-box này cũng được chọn để tránh các điểm bất động (fixed point).



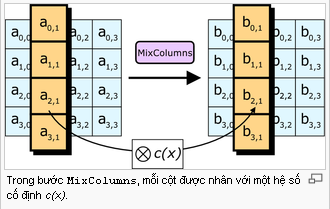
* Bước ShiftRows

Các hàng được dịch vòng một số vị trí nhất định. Đối với AES, hàng đầu được giữ nguyên. Mỗi byte của hàng thứ 2 được dịch trái một vị trí. Tương tự, các hàng thứ 3 và 4 được dịch 2 và 3 vị trí. Do vậy, mỗi cột khối đầu ra của bước này sẽ bao gồm các byte ở đủ 4 cột khối đầu vào. Đối với Rijndael với độ dài khối khác nhau thì số vị trí dịch chuyển cũng khác nhau.



* Bước MixColumns

Bốn byte trong từng cột được kết hợp lại theo một phép biến đổi tuyến tính khả nghịch. Mỗi khối 4 byte đầu vào sẽ cho một khối 4 byte ở đầu ra với tính chất là mỗi byte ở đầu vào đều ảnh hưởng tới cả 4 byte đầu ra. Cùng với bước ShiftRows, MixColumns đã tạo ra tính chất khuyếch tán cho thuật toán. Mỗi cột được xem như một đa thức trong trường hữu hạn và được nhân với đa thức *c*(*x*) = 3*x*3 + *x*2 + *x* + 2 (modulo *x*4 + 1). Vì thế, bước này có thể được xem là phép nhân ma trận trong trường hữu hạn.



### Thuật toán RSA

RSA là thuật toán mã hóa khóa bất đối xứng hay còn gọi là mã hóa khóa công khai. Thuật toán có hai [khóa](http://vi.wikipedia.org/wiki/Kh%C3%B3a_(m%E1%BA%ADt_m%C3%A3)): [khóa công khai](http://vi.wikipedia.org/wiki/Kh%C3%B3a_c%C3%B4ng_khai) và [khóa bí mật](http://vi.wikipedia.org/w/index.php?title=Kh%C3%B3a_b%C3%AD_m%E1%BA%ADt&action=edit&redlink=1). Mỗi khóa là những số cố định sử dụng trong quá trình mã hóa và giải mã.

#### Tạo khóa

* Chọn 2 số nguyên tố lớn p \, và q \, với p \ne q, lựa chọn ngẫu nhiên và độc lập.
* Tính: n = p q \,.
* Tính: giá trị hàm số Ơle  \phi(n) = (p-1)(q-1) \,.
* Chọn một số tự nhiên e sao cho 1 < e < \phi(n) \, và là  số nguyên tố cùng nhau với \phi(n) \,.
* Tính: d sao cho d e \equiv 1 \pmod{\phi(n)}.
* Khóa công khai bao gồm:

*n*: môđun, và

*e*: số mũ công khai

* Khóa bí mật bao gồm:

*n*: môđun, và

*d*: số mũ bí mật

#### Mã hóa

Dựa trên thừa nhận ta đã biết khóa công khai gồm n và e. Giả sử muốn mã hóa một số m < n. Ta tính c là bản mã hóa của m:

 c = m^e \mod{n} bằng phương pháp tính hàm mũ.

#### Giải mã

Từ c và khóa bí mật d. Ta tìm m lần lượt như sau:

m = c^d \mod{n}

Với c^d \equiv (m^e)^d \equiv m^{ed} \pmod{n}.

Do *ed* ≡ 1 (mod *p*-1) và *ed* ≡ 1 (mod *q*-1), (theo Định lý Fermat nhỏ) nên:

m^{ed} \equiv m \pmod{p}

Và m^{ed} \equiv m \pmod{q}

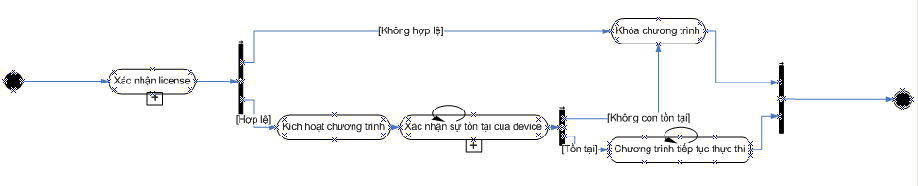
Do *p* và *q* là hai số nguyên tố cùng nhau, áp dụng định lý số dư Trung Quốc, ta có:

m^{ed} \equiv m \pmod{pq}.

hay:

c^d \equiv m \pmod{n}.

## Mô hình tổng thể



Trên đây là activity diagram thể hiện sự hoạt động của hệ thống. Đầu tiên là quá trình xác nhận license. Nếu thành công, chương trình sẽ được kích hoạt và thực thi. Ngược lại, chương trình luôn ở trạng thái bị khóa, không thể thao tác với các chức năng. Trong trường hợp ứng dụng đang thực thi, tùy theo chiến lược mà sau một khoảng thời gian sẽ có một quá trình xác nhận thiết bị Dongle có còn tồn tại hay không. Quá trình này được lặp lại trong suốt thời gian hoạt động của ứng dụng.

Ở cả hai quá trình xác nhận trên, việc quan trọng cần phải đảm bảo là dữ liệu truyền đi phải được mã hóa. Không những vậy, dữ liệu luôn phải được phát sinh động theo thời gian. Những chi tiết này sẽ được trình bày kỹ hơn ở phần sau.

# CHI TIẾT MÔ HÌNH

Trong chương này, ta sẽ tập trung vào hai quá trình quan trọng nhất của mô hình là xác nhận license và xác nhận sự tồn tại của thiết bị trong quá trình chạy ứng dụng. Cả hai quá trình này được mô tả chi tiết trong sơ đồ activity diagram đính kèm.

## Trao đổi khóa

### Giấy chứng nhận (Certificate)

Dùng để chứng minh khóa chung của một cá nhân nào đó. Về cơ bản, chỉ có các cơ quan có thẩm quyền (Certificate Authority) mới được phép cấp giấy chứng nhận. Tuy nhiên, với ứng dụng Dongle, ta có thể tự tạo giấy chứng nhận thông qua ID của thiết bị. Cũng giống như một giấy chứng nhận thông thường, giấy chứng nhận tự tạo phải là duy nhất. Điều này dễ dàng thực hiện được thông qua các ID duy nhất cho mỗi thiết bị.

### Bản tóm tắt (Digest)

Là một số nguyên kiểu int, có được bằng cách băm 1 chiều message gởi đi. Các thuật toán thường được dùng là MD5 128 bit và SHA 160 bit. Những thuật toán này đảm bảo các giá trị digest của message khác nhau thì khả năng giống nhau gần như bằng 0.

### Điều kiện kích hoạt quá trình:

* Ứng dụng được kích hoạt chạy ở host.
* Ứng dụng đang bị khóa và thiết bị được cắm vào host.

### Các bước thực hiện

Trong mô hình, quá trình mã hóa và giải mã đòi hỏi phải thống nhất khóa giữa host và thiết bị cho dù ta sử dụng thuật toán mã hóa đối xứng hay bất đối xứng. Việc trao đổi khóa sẽ được thực hiện bằng phương pháp RSA key exchange, được mô tả như sau:

Host: Xin chào!

Device: Chào host, đây là ***Giấy chứng nhận(Khóa chung)***.

Host: Hãy cho xem bằng chứng.

Device: (1)Tôi là device

(2)*Mã hóa bằng khóa riêng* {***Digest***[Tôi là device]}

Ở đây, một khi đã có giấy chứng nhận là host đã có được khóa chung. Host sẽ giải mã thông điệp mã hóa (2) của device và tính Digest[(1)] và so khớp. Nếu không khớp, device là giả mạo, kết thúc quá trình xác nhận. Ngược lại, host sẽ gởi khóa bí mật cho device:

Host: Mã hóa bằng khóa chung {Khóa bí mật}

Cuối cùng device sẽ có được khóa bí mật bằng cách giải mả thông qua khóa riêng của mình.

## Xác nhận license

### Điều kiện kích hoạt quá trình:

* Ứng dụng hoàn tất quá trình trao đổi khóa.

### Các bước thực hiện

* Host: Gởi một yêu cầu license đến thiết bị.
* Thiết bị: Nhận được yêu cầu, thực hiện load license từ vùng nhớ. Trường hợp không phát hiện thiết bị (không nhận được phản hồi), khóa chương trình ở host.
* Thiết bị: License sau khi được load sẽ trải qua 2 quá trình mã hóa lần lượt bằng khóa đối xứng và bất đối xứng.
* Thiết bị: Kết quả mã hóa được đóng thành từng block và truyền về host.
* Host: Nhận kết quả mã hóa, lần lượt trải qua 2 quá trình giải mã lần lượt bằng khóa bất đối xứng và đối xứng.
* Host: Kết quả giải mã sẽ được so khớp để xác định license thực. Nếu quá trình so khớp là hợp lệ, ứng dụng sẽ được thực thi đồng thời kích hoạt bộ đếm phục vụ cho quá trình xác nhận sự tồn tại của thiết bị về sau.

### Các vấn đề

Với các bước thực hiện trên, có một số điểm thừa nhận như sau:

* License là linh hoạt. Nó có thể là một chuỗi các ký tự, ký số tùy thuộc vào ứng dụng phát triển sau này. Tuy nhiên, khi lưu trữ, giá trị này là cố định trong mỗi thiết bị.
* Quá trình mã hóa – giải mã được thực hiện thông qua 2 lớp: AES và RSA.
* Quá trình truyền nhận theo block, ta có thể tự định nghĩa riêng một phương thức truyền nhận với các quy ước dữ liệu riêng.
* Quá trình so khớp đơn thuần là so sánh từng byte dữ liệu.

Các thừa nhận trên sẽ dẫn ta tới những vấn đề cần phải giải quyết như sau:

* Bất cứ lần truyền dữ liệu nào từ thiết bị lên host đều cùng có một nội dung duy nhất. Vì license là cố định và các thuât toán mã hóa cũng cho cùng một kết quả đối với cùng một bản dữ liệu. Vì vậy, cần làm cho dữ liệu này động bằng cách thêm một giá trị thời gian lấy từ host để trộn vào license.
* Việc so khớp không thể thực hiện trực tiếp bằng cách so từng byte mà phải thông qua một hàm trung gian để xác định đó có phải license thực hay không.

## Xác nhận tồn tại thiết bị

### Điều kiện kích hoạt quá trình

Ứng dụng đang thực thi và đã hoàn tất hợp lệ việc xác nhận license.

### Các bước thực hiện

Tại host

* Host tăng dần bộ đếm, khi đến thời điểm xác nhận đã quy ước trước, lấy giá trị hiện tại trong bộ đếm, tạm gọi là l1.
* Từ l1 thông qua một hàm tuyến tính G, phát sinh một chuỗi L2. Ở đây độ dài chuỗi do giá trị l1 thông qua G quyết định.
* Lấy phần tử cuối cùng của L2 để phát sinh ra chuỗi L3H theo cách tương tự bằng hàm tuyến tính G
* Đưa kết quả L2 lần lượt qua 2 thuật toán mã hóa đối xứng và bất đối xứng để có được L2’’
* Đóng thành block và chuyển cho thiết bị

Tại thiết bị

* Nhận L2’’, dùng 2 lần giải mã để chuyển về L2
* Phát sinh L3D từ L2 giống quá trình của host
* Mã hóa (2 lớp) và truyền L3D’’ theo block

Tại host

* Nhận L3D’’, giải mã 2 lớp để có được L3D
* So khớp L3D và L3H.
* Nếu hợp lệ, tiếp tục chương trình. Ngược lại khóa chương trình.

### Các vấn đề

#### Tràn số trong bộ đếm

Bộ đếm luôn được tăng lên ở mỗi nhịp đồng hồ sẽ dẫn đến nguy cơ bị tràn số. Ta giải quyết vấn đề này bằng cách khởi tạo lại giá trị này về 0 mỗi khi nó đạt đến một giới hạn nào đó.

#### Hàm G

Đây là hàm chuyển đổi một giá trị số k bất kỳ thành một chuỗi L. Đề xuất cách phát sinh như sau:

* Từ k dùng phép mod để tính toán số phần tử cho chuỗi L.

Len(L) = k mod H , với H là hằng số chọn trong lúc cài đặt

* Phát sinh Len(L) phần tử lần lượt bằng hàm:

li = a\*li-1 + c

Trong đó phần tử l0 đầu tiên sẽ được phát sinh từ k

Như vậy, một số k bất kỳ thông qua bước chuyển đổi trên sẽ được chuyển thành một chuỗi L.

## Giao tiếp giữa thiết bị và host

Trong mô hình này, một vấn đề quan trọng khác là xác lập cách giao tiếp giữa host và thiết bị. Vấn đề này đòi hỏi một số tính chất sau:

* Linh hoạt giữa các môi trường làm việc. Nghĩa là dễ dàng cài đặt, và sử dụng khi triển khai.
* Giao tiếp đảm bảo tính an toàn.
* Cấu trúc dữ liệu truyền phải dễ tinh chỉnh và mềm dẻo trong thay đổi cho phù hợp yêu cầu sử dụng

Với các yêu cầu đó, ta đề xuất hai cách giao tiếp như sau:

* Phát triển một driver usb riêng cho việc giao tiếp.

Cách này đòi hỏi phải xây dựng hoàn chỉnh mô hình giao tiếp giữa ứng dụng và firmware, thêm vào đó, ta phải thiết kế riêng một mô hình giao tiếp cho một driver đứng trung gian. Ưu điểm của mô hình này là sự chặt chẽ trong cấu trúc dữ liệu truyền được thông nhất ngay từ ban đầu. Tuy nhiên, chính vì điều đó mà ta khó thay đổi được cấu trúc sau này nếu dùng cho một yêu cầu đặc thù nào khác. Thêm vào đó, viêc cài đặt driver cũng đòi hỏi phải tương thích với môi trường chạy ứng dụng.

* Sử dụng thiết bị như HID (human interface device).

Theo cách này, môi trường Window sẽ tự động nhận thiết bị mà không cần phải xây dựng driver. Ưu điểm là sự linh hoạt trong việc định nghĩa dữ liệu truyền. Riêng về tính an toàn, không thể hoàn toàn kết luận được cách này có kém an toàn hơn hay không.

# TIẾN ĐỘ THỰC HIỆN

## Mô hình hiện tại

Về cơ bản, việc cài đặt được hiện thực hoàn toàn như mô tả. Tuy nhiên, do vấn đề thời gian, có một vài vấn đề chưa thực hiện được như sau:

* Thuật toán mã hóa và giải mã chỉ mới cài đặt phần khóa đối xứng AES
* Cách phát sinh trong hàm G chưa được hiện thực, hiện chỉ mới truyền đi chuỗi cố định trong quá trình xác nhận sự tồn tại thiết bị.
* Sử dụng cách giao tiếp HID

### Cách thực hiện

Sử dụng lại source code HID cho thiết bị tại trang chủ của Keil và thay đổi cho phù hợp với yêu cầu truyền từng block. (Ở source gốc, chỉ có thể truyền và nhận 1 byte). Về chi tiết cách sửa code sẽ viết trong một báo cáo sau.

Ứng dụng ở host là một chương trình đơn giản. Khi chạy, chương trình sẽ ở trạng thái khóa nếu không có thiết bị. Sau khi đã xác nhận thành công license, cứ sau 5 giây, thực hiện xác nhận tồn tại thiết bị một lần.

### Nhận xét

Mục đích ở bước đầu là xem xét cách giao tiếp và xử lý ở cả hai hướng.

* Giao tiếp giữa hai hướng thông qua HID thuận lợi. Có thể dễ dàng định nghĩa kiểu dữ liệu muốn truyền.
* Thuật toán AES nhanh, khả năng hồi đáp là tức thời.

## Công việc kế tiếp

Việc tiếp theo là hoàn tất kịch bản còn thiếu đối với cách tiếp cận HID như cách phát sinh dữ liệu truyền trong xác nhận tồn tại thiết bị, thêm cài đặt RSA, định nghĩa cấu trúc dữ liệu truyền nhận. Ngoài ra, có thể sẽ cần xem xét và hiện thực hóa kịch bản đối với hướng phát triển một driver usb. Song song đó là xem xét hiệu quả thực thi của mô hình (cả về tính bảo mật lẫn thời gian thực hiện) với nền tảng phần cứng thiết bị.