

**BAN CƠ YẾU CHÍNH PHỦ**

**HỌC VIỆN KỸ THUẬT MẬT MÃ**

**\*\*\***

****

**BÁO CÁO BÀI TẬP LỚN**

**PHÁT TRIỂN PHẦN MỀM TRONG THẺ THÔNG MINH**

Đề tài

***Xây dựng ứng dụng mã hoá giải mã dữ liệu***

***sử dụng thuật toán RSA***

**Giáo viên hướng dẫn: Cao Thanh Vinh**

**Sinh viên thực hiện:**

Trần Trọng Hữu

Lê Thị Quỳnh

Nguyễn Thị Ngọc Ánh

Phùng Thị Thuỳ

Đỗ Thị Thu Lệ

Hà Nội, 8/8/2020

# LỜI CẢM ƠN

Lời đầu tiên em xin gửi lời cảm ơn sâu sắc nhất đến cô ThS. Cao Thanh Vinh đã tận tâm, tận lực hướng dẫn, định hướng cho em, đồng thời, cũng đã cung cấp nhiều tài liệu và tạo điều kiện thuận lợi trong suốt quá trình học tập và nghiên cứu để em có thể hoàn thành tiểu luận này.

Em xin chân thành cảm ơn đến các thầy, cô trong Khoa Công nghệ thông tin, Trường Học Viện Kỹ Thuật Mật Mã cùng với ban lãnh đạo nhà trường đã nhiệt tình giảng dạy, cung cấp những trang thiết bị tốt nhất và truyền đạt những kiến thức, kinh nghiệm qúy giá trong suốt quá trình học tập rèn luyện tại trường.

Em xin gửi lời cảm ơn đến các bạn học viên lớp CT1, đã đồng hành cùng em trong suốt quá trình học tập. Cảm ơn gia đình, bạn bè đã quan tâm và động viên giúp em có nghị lực phấn đấu để hoàn thành tốt tiểu luận này. Do kiến thức và thời gian có hạn nên tiểu luận sẽ không tránh khỏi những thiếu sót nhất định.

Em rất mong nhận được những sự góp ý quý báu của thầy cô, bạn bè. Một lần nữa xin gửi lời cảm ơn chân thành và sâu sắc.

Hà Nội, 8 tháng 8 năm 2020

Nhóm sinh viên thực hiện.

MỤC LỤC

[LỜI CẢM ƠN 2](#_Toc47826406)

[DANH MỤC HÌNH VẼ 5](#_Toc47826407)

[LỜI MỞ ĐẦU 6](#_Toc47826408)

[CHƯƠNG 1: GIỚI THIỆU VỀ THẺ THÔNG MINH 7](#_Toc47826409)

[1.1. Lịch sử phát triển thẻ thông minh. 7](#_Toc47826410)

[1.2. Cấu tạo và phân loại thẻ thông minh 9](#_Toc47826411)

[1.3. Ưu nhược điểm của thẻ thông minh 13](#_Toc47826412)

[1.4. Thách thức trong việc phát triển ứng dụng thẻ thông minh 16](#_Toc47826413)

[1.5. Các hình thức tấn công trên thẻ thông minh 17](#_Toc47826414)

[CHƯƠNG 2: CÔNG NGHỆ JAVACARD 20](#_Toc47826415)

[2.1. Giới thiệu JavaCard 20](#_Toc47826416)

[2.2. Kiến trúc JavaCardfe 21](#_Toc47826417)

[2.3. Tập ngôn ngữ JavaCard 22](#_Toc47826418)

[2.4. Máy ảo để chạy JavaCard 23](#_Toc47826419)

[2.5. Cài đặt JavaCard và chương trình cài đặt trên thiết bị (Off-Card) 25](#_Toc47826420)

[2.6. Môi trường chạy JavaCard 27](#_Toc47826421)

[2.7. API JavaCard 29](#_Toc47826422)

[2.8. Package và quy ước đặt tên Applet 31](#_Toc47826423)

[2.9. Java Card Applet 32](#_Toc47826424)

[2.9.1. Tiến trình phát triển Applet 32](#_Toc47826425)

[2.9.2. Cài đặt Applet 34](#_Toc47826426)

[2.10. Phương thức truyền nhận, trao đổi dữ liệu 36](#_Toc47826427)

[CHƯƠNG 3: HỆ MÃ RSA, VÀ THUẬT TOÁN MÃ HÓA AES 39](#_Toc47826428)

[3.1. Thuật toán RSA 39](#_Toc47826429)

[3.2. Mã hóa RSA CRT 41](#_Toc47826430)

[3.3. Thuật toán AES 42](#_Toc47826431)

[CHƯƠNG 4: ỨNG DỤNG CÔNG NGHỆ JAVACARD XÂY DỰNG ỨNG DỤNG THẺ KHÁM BỆNH TRÊN NETBEAN IDE 44](#_Toc47826432)

[4.1 Giao tiếp giữa javacard applet và netbean 44](#_Toc47826433)

[4.1.1 Java Card Runtimes 44](#_Toc47826434)

[4.1.2 Apdu in out library 44](#_Toc47826435)

[4.2 Thiết kế phần mềm 46](#_Toc47826436)

[4.2.1 Biểu đồ phân rã chức năng 46](#_Toc47826437)

[4.2.2 Biểu đồ hoạt động 47](#_Toc47826438)

[4.3. Thiết kết giao diện 50](#_Toc47826439)

[4.4 Phát triển Applet 53](#_Toc47826440)

[**4.4.1. Hoạt động mã hoá** 53](#_Toc47826441)

[**4.4.2.** **Giải mã dữ liệu** 53](#_Toc47826442)

# DANH MỤC HÌNH VẼ

[Hình 1.1 hình ảnh của thẻ thông minh 8](#_Toc47821498)

[Hình 1.2 Cấu trúc của thẻ thông minh 10](#_Toc47821499)

[Hình 1.3 Thẻ thông minh và đầu đọc thẻ 12](#_Toc47821500)

[Hình 1.4 Thẻ không tiếp xúc 13](#_Toc47821501)

[Hình 2.1 Kiến trúc tổng quát của công nghệ Javacard 23](#_Toc47821502)

[Hình 2.2 Các thuộc tính mà thư viện hỗ trợ 24](#_Toc47821503)

[Hình 2.3 Máy ảo của javacard. 24](#_Toc47821504)

[Hình 2.4 Trình cài đặt JavaCard và chương trình cài đặt ngoài thẻ 28](#_Toc47821505)

[Hình 2.5 Kiến trúc hệ thống trên thẻ 29](#_Toc47821506)

[Hình 2.6 Tiến trình phát triển Applet 34](#_Toc47821507)

[Hình 2.7 Trao đổi thông tin giữa ứng dụng trên thẻ và ứng dụng trên thiết bị đầu cuối 37](#_Toc47821508)

[Hình 2.8 Mã trạng thái phản hồi 39](#_Toc47821509)

[Hình 4.1. Cấu hình của thiết bị card ảo. 42](#_Toc47821510)

[Hình 4.2. Biểu đồ phân rã chức năng 46](#_Toc47821511)

[Hình 4.3. Biểu đồ luồng hoạt động thẻ khi chưa có dữ liệu 45](#_Toc47821512)

[Hình 4.4. Giao diện nhập thông tin bệnh nhân 48](#_Toc47821513)

[Hình 4.5. Giao diện APDU lệnh 49](#_Toc47821514)

[Hình 4.6. Giao diện hiển thị thông tin bệnh nhân 50](#_Toc47821515)

# LỜI MỞ ĐẦU

Với sự phát triển nhanh chóng của xã hội, các thiết bị thông minh ra đời ngày càng nhiều, giúp nâng cao và thay đổi cuộc sống của con người trở nên nhanh chóng, tiện lợi hơn, thông minh hơn, một trong số các thiết bị thông minh đó là thẻ thông minh (Smart card). Kèm theo đó là sự phát triển của công nghệ bán dẫn cho phép các nhà sản xuất chip tạo ra những con chip hay thẻ thông minh ngày càng nhỏ gọn cùng với sức mạnh tính toán cao. Tuy nhiên việc có quá nhiều nhà sản xuất chip, công việc phát triển ứng dụng cho thẻ thông minh gặp khó khăn về sự tương thích. Do đó nhu cầu về một nền tảng chung bên trong chip được đặt ra, công nghệ Java Card được phát triển để phục vụ mục đích này. Với việc tạo ra một môi trường ảo chung trên tất cả các hệ điều hành hỗ trợ JavaCard, công nghệ này đã giúp cho việc phát triển ứng dụng chip trở nên dễ dàng giúp tiết kiệm thời gian nghiên cứu phát triển.

Trong phạm vi môn học: *Phát triển phần mềm trong thẻ thông minh*, nhóm chúng em nhận được đề tài: “**Xây dựng ứng dụng mã hóa, giải mã dữ liệu sử dụng thuật toán RSA**”. Với đề tài nghiên cứu trên nhóm chúng em đưa ra giải pháp xây dựng một ứng dụng lưu trữ thông tin cơ bản của một bệnh nhân.

Hướng nghiên cứu của đề tài:

* Nghiên cứu công nghệ JavaCard
* Nghiên cứu hệ mật RSA và AES
* Nghiên cứu triển khai một Applet dựa trên Netbean IDE 8.2

Ngoài phần mở đầu, kết luận, nội dung tiểu luận bao gồm các chương sau:

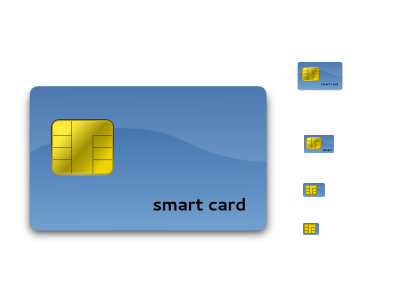
* Chương 1: **Giới thiệu về thẻ thông minh.** Chương này trình bày khái quát về thẻ thông minh.
* Chương 2: **Công nghệ JavaCard**. Chương này giới thiệu các nội dung cơ bản của công nghệ Javacard.
* Chương 3: **Hệ mã RSA, và thuật toán mã hóa AES.** Chương này trình bày về hệ mã bất đối xứng RSA, và giới thiệu về thuật toán mã hóa đối xứng AES.
* Chương 4: **Ứng dụng công nghệ JavaCard xây dựng ứng dụng thẻ khám bệnh trên Netbean IDE.** Chương này trình bày chi tiết về cách triển khai một ứng dụng javacard trên Netbean, chi tiết về ứng dụng khám chữa bệnh và cách mã hóa, giải mã dữ liệu dựa trên thuật toán RSA, AES.

# CHƯƠNG 1: GIỚI THIỆU VỀ THẺ THÔNG MINH

Thẻ thông minh đang được ứng dụng rộng rãi tại Việt Nam trong nhiều lĩnh vực như viễn thông, ngân hàng, thương mại điện tử, điều khiển tự động, kiểm soát người và phương tiện… Các ứng dụng của thẻ thông minh rất thiết thực và tích hợp phần mềm điều khiển bởi ưu điểm vượt trội về khả năng lưu trữ, xử lý thông tin và bảo mật dữ liệu. Chương I trình bày cái nhìn tổng quan về thẻ thông minh.

## Lịch sử phát triển thẻ thông minh.

Có hai ý tưởng chính đã dẫn đến sự phát triển của thẻ thông minh. Ý tưởng đầu tiên xuất hiện bởi Tiến sĩ Kunitaka Arimura đến từ Nhật Bản, ông có thiết kế tích hợp dữ liệu lưu trữ và logic số học vào một miếng silicon, ông đã nộp bản quyền cho ý tưởng vào năm 1970. Ý tưởng thứ hai là kỹ sư người Đức Helmut Grӧttrup và đồng nghiệp là Jϋrgen Dethloff, họ đã nộp bản quyền năm 1968. Bằng sáng chế thẻ chip tự động này được công bố vào cuối năm 1982. Năm 1974, Roland Moreno – một nhà phát minh của Pháp, đã gắn chip lên một tấm nhựa và cấp bằng sáng chế về thẻ nhớ và thiết bị đọc nó, được đặt tên là thẻ thông minh. Moreno đã thành lập công ty Innovatron để bán ý tưởng, Moreno được biết như là cha đẻ của mạch vi xử lý (Microchip).



Hình 1.1 hình ảnh của thẻ thông minh

Năm 1977, ba nhà sản xuất thương mại, Bull CP8, SGS Thomson và Schlumberger đã bắt đầu phát triển các sản phẩm của thẻ thông minh. Năm 1978, Bull đăng ký bằng sáng chế về bộ vi xử lý một chip tự lập trình được (SPOM-self Programmable One-chip Microcomputer). Tháng 3 năm 1979, Michel Ugon của tập đoàn Bull là người đầu tiên thiết kế bộ vi xử lý hoạt động, được biết đến như là CP8 của Bull. Nó chứa bộ nhớ lập trình 1KB, bộ vi xử lý 6805 do Motorola sản xuất. Đây có thể coi là thẻ thông minh đầu tiên kết hợp sức mạnh của bộ vi xử lý và bộ nhớ có khả năng đưa ra quyết định dựa trên nhu cầu của người dùng để sửa đổi, thêm, truy xuất hoặc xóa dữ liệu được lưu trữ. Thẻ này là một thiết kế hai chip, trong đó bộ nhớ và bộ vi xử lý là hai đơn vị riêng biệt, được chứng minh là một giải pháp không an toàn. Năm 1980 cho phép tích hợp tất cả các mạch vào trong một con chip. Đến năm 1981 những chiếc thẻ thông minh đã có nền tảng ở nhiều nước Tây Âu, một số các ngân hàng Châu Âu đồng ý thành lập một cơ quan quản lý mới cho phát triển thẻ, ứng dụng và tiêu chuẩn. Tổ chức này bao gồm các tổ chức tài chính từ Bỉ, Anh, Đan Mạch, Áo, Hà Lan, và liên minh cũ các ngân hàng Pháp.

Sự bùng nổ thẻ thông minh bắt đầu vào trong thập niên 90, khi có sự xuất hiện của SIM dùng trong thiết bị điện thoại di động GSM ở châu Âu, cùng với việc mạng di động mở rộng khắp châu Âu, thẻ thông minh ngày càng trở nên thông dụng. Năm 1983 cộng đồng viễn thông yêu cầu một hệ thống điện thoại trả tiền tốt hơn do tỷ lệ gian lận và hack gia tăng đối với điện thoại công cộng sử dụng đồng xu. Hệ thống thanh toán thẻ điện thoại trả tiền thông minh đã phát triển. Schlumberger đã bắt đầu cài đặt hàng ngàn chiếc điện thoại trả tiền bằng thẻ thông minh trên khắp lục địa. Đến cuối năm họ đã cài đặt được 160.000 chiếc điện thoại. Vào năm 1984, một trong những triển khai lớn nhất của thẻ thông minh đã diễn ra tại Pháp. Ngành ngân hàng Pháp đã quyết định làm cho thẻ thông minh trở thành tiêu chuẩn cho thẻ tín dụng và thẻ ghi nợ.

Năm 1993, ba liên minh thẻ lớn nhất thế giới là Europay, MasterCard, Visa (EMV) thỏa thuận cùng hợp tác để xây dựng nên kỹ thuật cho việc dùng thẻ thông minh trong các thẻ thanh toán ở hai loại là thẻ tài khoản và thẻ tín dụng. Phiên bản đầu tiên của hệ thống EMV này được công bố vào năm 1994. Năm 1998, phiên bản tin cậy hơn là EMVco, chịu trách nhiệm bảo trì lâu dài hệ thống này. Mục đích của công ty EMVco là đảm bảo với các tổ chức tài chính và các đại lý rằng các chuẩn kỹ thuật dù phát triển nhưng vẫn phải giữ được tương thích với phiên bản 1998. Hiện nay, thẻ thông minh đã và đang sử dụng cho rất nhiều lĩnh vực như làm thẻ chứng minh nhân dân, hộ chiếu điện tử, thẻ thanh toán. Tại Việt Nam thẻ thông minh hiện đang sử dụng nhiều nhất là thẻ tín dụng và sim điện thoại.

## Cấu tạo và phân loại thẻ thông minh

Thẻ thông minh được ví như một máy tính di động lưu trữ chương trình dữ liệu. Ngày nay, thẻ thông minh đang được ứng dụng rộng rãi tại Việt Nam trong nhiều lĩnh vực khác nhau như giao thông công cộng, viễn thông, ngân hàng, thương mại điện tử, điều khiển tự động, điều khiển người và phương tiện, y tế giáo dục,v.v…Các ứng dụng của thẻ thông minh rất thực tế, dễ triển khai và tích hợp phần mềm điều khiển bởi ưu thế nổi bật về khả năng lưu trữ, xử lý thông tin, bảo mật dữ liệu tốt, khả năng tích hợp linh hoạt, việc sử dụng thẻ thông minh đem lại nhiều lợi ích cho người sử dụng. Để có cái nhìn tổng quan về thẻ thông minh, cần tìm hiểu cấu tạo, nguyên lý hoạt động, các loại thẻ thông minh trên thị trường, cách thức phát triển phần mềm quản lý thẻ thông minh.

1. *Cấu tạo thẻ thông minh:*

Thẻ thông minh thông thường có kích thước cỡ một thẻ tín dụng và được làm bằng nhựa, thường là PVC (Polyvinyl chloride – là thẻ nhựa cứng và không có mùi) đôi khi là ABS (Acrylonnitrile, Butadiene, Styrene – là một thẻ nhựa chịu được sự va đập mạnh), thẻ có thể chứa một ảnh 3 chiều tránh lừa đảo. Kích thước theo chuẩn ID1(ISO/IEC/7810) quy định là 85,60x53,98 mm hoặc chuẩn ID-000 kích thước 25x15 mm, có bề dày mặt thẻ là 0,76 mm. Thẻ được gắn một bộ mạch tích hợp cho phép giao tiếp với hệ thống, tính toán mã hóa đảm bảo tính toàn vẹn dữ liệu. Dung lượng bộ nhớ trên thẻ thông minh khá lớn từ 64KB - 128KB (tương đương 65.536 đến 131.072 ký tự) Dữ liệu trên thẻ được truyền đến hệ thống quản trị trung tâm nhờ vào các thiết bị đọc thẻ, chẳng hạn máy đọc thẻ, ATM v.v. Hình 1.2 thể hiện cấu trúc các lớp của thẻ thông minh.



Hình 1.2 Cấu trúc của thẻ thông minh

1. *Phân loại thẻ thông minh:* Có hai cách phân loại thẻ thông minh dựa trên công nghệ chip hay phương thức đọc dữ liệu.

**\* Phân loại dựa trên công nghệ chip**: Theo công nghệ chip được chia làm hai loại là thẻ chip nhớ (memory chip) và thẻ chip vi xử lý (microprocessor chip) được gắn trên bề mặt thẻ.

- Thẻ chip nhớ bao gồm hai thành phần chính là thẻ nhớ cho phép có thể truy cập, giao thức truyền thông. Ưu điểm của thẻ này là dễ sản xuất, dễ sử dụng, nhược điểm là hạn chế về bộ nhớ và tính bảo mật không cao

- Thẻ vi xử lý (microprocessor chip) được cấu tạo bởi ba loại bộ nhớ, một bộ vi xử lý (CPU – Central Processing Unit), một bộ đồng xử lý mã hóa (Cryto coprocessor) và một giao diện thông tin (communication interface). Chức năng của CPU là điều khiển các bộ phận khác, xử lý thông tin và thực hiện các phép tính. Cấu tạo của CPU rất đa dạng, nhưng nói chung gồm 1 bộ xử lý (control unit) đảm nhận những chu trình cơ bản của CPU như đọc một chỉ thị và thực hiện nó, giải mã (decoding), lưu trữ (stocking), đảm nhận chức năng ALU (arithmetic and logic unit), quản lý thanh ghi, quản lý bộ nhớ (registers, RAM, ROM). ROM dùng để lưu trữ mã máy (code), dữ liệu (data), và chỉ có thể đọc, chứ không thể thay đổi nội dung. Thông tin trong ROM vẫn nguyên vẹn, ngay cả khi chúng ta ngắt card. Trong ngành thẻ thông minh, ROM được dùng để lưu trữ những ứng dụng sẽ được thực hiện bởi bộ vi xử lý. Dung lượng của ROM vào khoảng 256KB là tối đa, do thiếu không gian lắp đặt. EEPROM (Electrically Erasable Programmable Read Only Memory) giống ROM ở chỗ là thông tin lưu trữ vẫn nguyên vẹn, ngay cả khi card bị ngắt khỏi nguồn năng lượng. EPPROM có thêm 1 lợi thế là có thể cùng lúc ở mode đọc hoặc ghi. Giống ROM, dung lượng EPPROM vào khoảng vài trăm KB, do thiếu không gian. Ngày nay, sự xuất hiện của những công nghệ mới như bộ nhớ Flash, hoặc RAM sắt điện (FeRAM) với thời gian đọc, ghi, xóa ngắn hơn nhiều và kích thước của bit nhớ cũng nhỏ hơn điều đó đã tăng dung lượng nhớ của thẻ thông minh lên rất nhiều. RAM là một loại bộ nhớ nhanh và không vĩnh cửu (sẽ bị xóa khi ngắt khỏi nguồn năng lượng). RAM chỉ được sử dụng bởi bộ vi xử lý, các yếu tố bên ngoài không thể truy cập vào RAM. RAM khá đắt, và cũng chiếm nhiều không gian, nên thường dung lượng không nhiều, khoảng vài Kb. Bộ đồng xử lý mã hóa (Crypto coprocessor) để đáp ứng nhu cầu hiệu năng, một vài loại thẻ thông minh được trang bị thêm một chip điện tử. Chip này được thiết kế đặc biệt để có thể thực hiện các phép tính số học trên những số rất lớn (vài trăm đến vài nghìn bits) một cách tối ưu. Chức năng của chip này là để thực hiện các hàm mã hoá xuất hiện trong các giao thức (protocol) của thẻ thông minh. Thời kỳ đầu, chip mã hoá chỉ được trang bị trên 1 số loại thẻ thông minh vì đắt. Nhưng hiện nay chúng ta có thể tìm thấy thành phần này trên hầu như tất cả các loại thẻ thông minh.

**\* Phân loại dựa trên phương thức đọc dữ liệu**: chia làm 3 loại là thẻ tiếp xúc, thẻ không tiếp xúc và thẻ lưỡng tính

- Thẻ tiếp xúc: là loại thẻ có một diện tích tiếp xúc thường dễ nhận diện bởi có gắn con chip (màu vàng hoặc bạc) trên thân thẻ, tiếp điểm đó có diện tích khoảng 1cm2 , được chia thành các phần riêng biệt gồm đầu vào, đầu ra dữ liệu, tín hiệu reset (phục hồi trạng thái ban đầu của thẻ), tín hiệu xung đồng hồ, chân điện áp. Con chip này nhỏ nhưng có các chức năng không khác gì một chiêc máy vi tính, muốn đọc, ghi thông tin thì bề mặt chip phải tiếp xúc trực tiếp với đầu đọc thẻ. Loại thẻ này được sử dụng nhiều trong tài chính và truyền thông (ví dụ sim điện thoại) vì ưu điểm giá cả rẻ, đáp ứng nhiều tiêu chuẩn về công nghệ, độ bảo mật cao. Khi được đưa vào máy đọc, chip trên thẻ sẽ giao tiếp với các tiếp điểm điện tử cho phép đọc các thông tin từ chip và viết thông tin lên nó. Thẻ thông minh loại này không có pin, năng lượng làm việc sẽ được cấp trực tiếp từ máy đọc thẻ.

Hình 1.3 Thẻ thông minh và đầu đọc thẻ

Các chuẩn ISO/IEC 7816 và ISO/IEC 7810 qui định:

* + Hình dạng và kích thước vật lý.
  + Vị trí và hình dạng của các tiếp điểm điện tử.
  + Các đặc tính điện.
  + Các giao thức thông tin, bao gồm định dạng của các lệnh gửi đến thẻ và các đáp ứng từ thẻ.
  + Độ tin cậy của thẻ.
  + Chức năng

- Thẻ không tiếp xúc: Là loại thẻ mà chip trên nó liên lạc với máy đọc thẻ thông qua công nghệ sóng vô tuyến RFID (Radio Frequency Identification) với tốc độ trao đổi dữ liệu từ 106 đến 848 kbit/s, thân thẻ chứa chip và đường dây ăngten được dấu ngầm. Ăngten đi vòng quanh thẻ, nó có nhiệm vụ làm trung gian nhận hoặc phát sóng radio giữa đầu đọc thẻ và chip trên thẻ. Trong thẻ có một cuộn cảm có khả năng dò tín hiệu vô tuyến trong một dải tần nhất định, chỉnh lưu tín hiệu và dùng nó để cung cấp năng lượng hoạt động cho chip trên thẻ, khoảng cách giao tiếp giữa đầu đọc thẻ và máy khoảng 10cm. Tốc độ xử lý của thẻ không tiếp xúc nhanh hơn so với thẻ tiếp xúc, vì vậy thẻ không tiếp xúc thường được ứng dụng tại những nơi cần phải xử lý nhanh như kiểm soát phương tiện công cộng, xe bus, thẻ ra vào…thẻ không tiếp xúc đắt hơn thẻ tiếp xúc, tuy nhiên độ bảo mật thông tin không an toàn bằng thẻ tiếp xúc. Về mặt cấu tạo thì thẻ không tiếp xúc khác với thẻ tiếp xúc là con chip quản lý thông tin không nằm lộ trên thẻ mà ẩn bên trong thẻ.



Hình 1.4 Thẻ không tiếp xúc

- Thẻ lưỡng tính: là thẻ kết hợp các đặc điểm của thẻ tiếp xúc và thẻ không tiếp xúc. Dữ liệu được truyền hoặc bằng phương pháp tiếp xúc trực tiếp thẻ với đầu đọc hoặc qua tín hiệu vô tuyến. Thẻ lưỡng tính đắt hơn rất nhiều so với 2 loại thẻ trên. Đầu đọc thẻ thông minh là đầu đọc dùng cho việc giao tiếp với thẻ, dữ liệu và điện năng được truyền trực tiếp hoặc gián tiếp qua công nghệ RFID từ thẻ vào máy đọc. Đầu đọc thẻ dễ dàng tích hợp vào các hệ thống khác nhau thông qua thiết bị đầu cuối. Tùy vào công nghệ sử dụng, có 2 dạng cổng kết nối là USB và COM. Nếu sử dụng kết nối USB tốc độ truyền tín hiệu đạt 12 Mbps (High speed), điện áp cung cấp thông thường 5V, 200mA.

**Các ứng dụng tiêu biểu của thẻ thông minh:**

* Định danh: Đối với các hệ thống cần xác nhận định danh được phép truy cập hệ thống như: Mạng viễn thông di động, tài khoản ngân hàng, chứng minh nhân dân điện tử, hộ chiếu điện tử hay hệ thống quản lý truy cập (Access Control) thì TTM được đại diện cho quyền truy cập các hệ thống này.
* Lưu trữ: Khả năng lưu trữ an toàn trên thẻ smartcard, cho phép lưu trữ những thông tin thuộc về chủ thẻ như thông tin y tế, thông tin cá nhân, chứng chỉ điện tử (thẻ bảo hiểm y tế, giấy phép lái xe điện tử, v.v...).
* Xác thực Offline: Ngoài các ứng dụng phổ biến nói trên, thẻ thông minh còn được dùng để kiểm tra tính xác thực thẻ thành viên không yêu cầu kết nối hệ thống trung tâm. Thẻ SAM card (Security Application Module) là một dạng của ứng dụng này. SAM card có vai trò như một cảnh sát giao thông kiểm tra người lái xe xuất trình bằng lái tại xa lộ mà không có máy tính hay kết nối cơ sở dữ liệu trung tâm. Khi đó, thẻ SAM và thẻ của người cần kiểm tra sẽ kiểm tra lẫn nhau (xác thực chéo) để kiểm tra tính trung thực trước khi thực hiện những nghiệp vụ tiếp theo. Thẻ SAM còn có khả năng đa dạng khóa (Diversify Key) đảm bảo an toàn và bảo mật trong mỗi phiên giao dịch.

## Ưu nhược điểm của thẻ thông minh

* **Ưu điểm của thẻ thông minh**: Thẻ thông minh với cấu tạo chip có nhiều ưu điểm hơn so với các loại thẻ từ khác. Ưu điểm của thẻ thông minh là:
* *Thẻ thông minh được ứng dụng được trong nhiều lĩnh vực*: Thẻ thông minh ứng dụng tiện lợi trong nhiều lĩnh vực như thẻ công dân, hộ chiếu điện tử, thẻ y tế - lưu trữ thông tin cần thiết như nhóm máu, sinh trắc học của người chủ thẻ, thanh toán lương - thẻ tín dụng (ATM), sinh hoạt phí hàng tháng như thẻ đổ xăng, SIM cho điện thoại di động, thẻ truyền hình cho các kênh phải trả tiền, các thẻ dùng cho thu phí giao thông tự động, thanh toán tiền xe bus, tàu, chi phí du lịch, nhà hàng quán ăn, cửa hàng buôn bán, trung tâm chăm sóc spa, bãi đỗ xe, siêu thị... Thẻ thông minh cũng dùng như ví điện tử dùng để trả tiền tại các trạm đỗ xe và các máy bán hàng tự động. Ngoài ra còn ứng dụng trong lĩnh vực an ninh cho máy tính, hệ thống mã hóa dữ liệu trên đĩa cứng có thể dùng thẻ thông minh để giữ các khóa mã bảo mật
* *Tính bảo mật cao:* Tính bảo mật là ưu điểm nổi bật nhất của thẻ thông minh bởi các thành phần vật lý của con chip đều ở dạng siêu nhỏ và chúng đều có khả năng chống lại các tấn công vật lý. Còn về khả năng tấn công hay tìm cách đọc nội dung dữ liệu lưu trong thẻ bằng phần mềm đã được hệ điều hành toàn quyền điều khiển. Hệ điều hành thẻ đều phải tuân theo các tiêu chuẩn ISO-7816 với nhiều mức bảo vệ truy cập nhiều cấp, nên rất khó để tấn công dữ liệu theo con đường này. Ngoài ra, trong thẻ còn hỗ trợ các thuật toán mã hóa, các cơ chế chống nhân bản thẻ (anti-cloning) hay an toàn trong quá trình trao đổi dữ liệu, hay những thẻ đáp ứng chuẩn EMV giúp ngăn ngừa giả mạo, thẻ EMV có một bộ vi xử lý bên trong được gắn vào con chip trên thẻ, chúng bao gồm các khóa mật mã để chứng minh thẻ là bản gốc nên nếu chủ thẻ không may đánh mất thẻ thì ngân hàng chỉ cần khóa chip là toàn bộ các giao dịch với thẻ là không thể thực hiện được. Thẻ chip có thể làm giảm đáng kể việc đánh cắp hoặc sao chép dữ liệu và thông tin của chủ thẻ so với thẻ từ.
* *Khả năng lưu trữ thông tin lớn* Thẻ thông minh được ví như một máy tính cá nhân thu nhỏ bởi nó có thể lưu trữ 10 một lượng thông tin rất lớn về cá nhân tổ chức. Việc quản lý các thông tin này cũng rất dễ dàng vì không cần phải tích hợp thêm phần mềm nào. Mặt khác thông tin lưu trên thẻ thông minh có thể dễ dàng thay đổi, xóa hoặc thêm bớt khi cần.
* *Khả năng xử lý thông tin nhanh* Với công nghệ chip điện tử thẻ thông minh xử lý thông tin rất nhanh. Chính vì thế thẻ thông minh thường được ứng dụng trong những giao dịch yêu cầu về thời gian giao dịch nhanh như thanh toán phí giao thông, thẻ gửi xe…
* *Có nhiều dịch vụ hỗ trợ người dùng và đơn giản hóa thủ tục* Thẻ thông minh cho phép thực hiện các giao dịch kinh doanh một cách hiệu quả theo một cách chuẩn mực, linh hoạt và an ninh mà trong đó con người ít phải can thiệp vào. Ngoài ra thẻ thông minh với giao tiếp không cần tiếp xúc đã trở nên ngày càng phổ biến trong các ứng dụng thanh toán và mua vé, điển hình là lời giải cho bài toán bán vé vận tải công cộng. Thẻ thông minh sẽ giúp đơn giản hóa thủ tục và thời gian vận chuyển bằng các phương tiện công cộng.
* *Sử dụng trên phạm vi quốc tế* Hiện nay các loại thẻ thông minh visa, master card đều có thể sử dụng trên phạm vi quốc tế, người dùng có thể thanh toán mua hàng online, phục vụ việc du lịch, học tập… người dùng có thể đi bất kỳ đâu mà không cần mang quá nhiều tiền.
* **Hạn chế của thẻ thông minh**
* *Dễ bị mất, dễ hư hỏng* Giống như thẻ tín dụng thì thẻ thông minh nhỏ, nhẹ và có thể dễ dàng bị mất nếu người đó không có trách nhiệm. Không giống như thẻ tín dụng, thẻ thông minh có thể được sử dụng vào nhiều mục đích có thể là cùng một thẻ do đó việc mất thẻ thông minh có thể gây nhiều bất lợi cho người sở hữu, ví dụ như nếu mất đi một thẻ debit (thẻ ghi nợ), thẻ xe buýt, chìa khóa văn phòng, v.v… điều này gây nên sự bất tiện nghiêm trọng. Ngoài ra một nhược điểm nữa của thẻ thông minh là dễ hư hỏng. Thẻ nhựa mà chip đặt trên nó là khá dẻo, dễ uốn, và do đó chip càng lớn thì càng dễ bị gãy. Thẻ thông minh thường được bỏ trong ví đây là một môi trường khá khắc nghiệt đối với chip điện tử.
* *Vấn đề an toàn thẻ thông minh*: Không phải tất cả các thẻ thông minh đều an toàn. Visa và MasterCard đã phát triển một tiêu chuẩn mới mục đích đưa toàn bộ ngành công nghiệp đạt được tiêu chuẩn mã hóa. Thẻ thông minh dùng để xác nhận khách hàng là một trong những cách an ninh nhất, có thể dùng trong những ứng dụng như giao dịch ngân hàng qua internet, nhưng mức độ an toàn không thể đảm bảo 100%. Trong trường hợp giao dịch ngân hàng qua internet, nếu máy tính bị nhiễm bởi các phần mềm xấu, mô hình an ninh sẽ bị phá vỡ. Phần mềm xấu có thể được viết đè lên thông tin (cả thông tin đầu vào từ bàn phím và thông tin đầu ra màn hình) giữa khách hàng và ngân hàng. Nó có thể sẽ sửa đổi giao dịch mà khách hàng không biết. Hiện nay có những phần mềm xấu chẳng hạn như Trojan, Silentbanker. Các ngân hàng như Fortis Dexia ở Bỉ dùng một thẻ thông minh chung với một máy đọc thẻ không nối mạng nhằm giải quyết vấn đề trên. Khách hàng nhập một thông tin đánh giá từ trang web của ngân hàng, PIN của họ, và tổng số tiền giao dịch vào một máy đọc thẻ, máy đọc thẻ sẽ trả lại một chữ ký 8 chữ số. Chữ ký này sẽ được khách hàng nhập bằng tay vào PC và được kiểm chứng bởi ngân hàng. Bên cạnh việc chạy đua kỹ thuật cũng là sự thiếu hẳn một chuẩn thống nhất về chức năng và an ninh của thẻ thông minh. Để giải quyết vấn đề này, dự án ERIDANE đã được khởi động bởi The Berlin Group để phát triển một khung chức năng và an ninh cho những thiết bị bán lẻ đầu cuối dùng thẻ thông minh
* *Tăng nguy cơ phạm tội* Khi được sử dụng đúng mục đích nhận dạng nó làm cho công việc của nhân viên hành pháp và các chuyên gia chăm sóc sức khoẻ dễ dàng hơn. Tuy nhiên, đối với bọn tội phạm tìm kiếm hay đánh cắp thông tin thì chúng dựa trên số lượng thông tin mà nó có thể chứa trên thẻ, những tên tội phạm có thể lấy các thông tin bất hợp pháp trên thẻ để phục vụ các mục đích cá nhân của chúng ví dụ như các tên tội phạm khi lấy cắp được thông tin thẻ chúng có thể thực hiện giao dịch bất hợp pháp hoặc rao bán thông tin bất hợp pháp của chủ thẻ v.v…
* *Rủi ro về quyền riêng tư* Dùng thẻ thông minh cho giao thông công cộng cũng có một chút rủi ro về quyền tự do cá nhân, bởi vì với hệ thống như vậy thì người quản lý giao thông có thể dò theo hành trình của cá nhân. Ở Phần Lan, bộ phận bảo vệ Dữ Liệu Ombudsman cấm người quản lý giao thông của YTV thu thập các thông tin như vậy, mặc dù trong hợp đồng với YTV người chủ thẻ có quyền yêu cầu YTV cung cấp cho họ lịch trình đi mà YTV đã tính tiền cho họ. Những thông tin về lịch trình từng được dùng trong việc truy tìm thủ phạm trong vụ đánh bom Myamanni.
* *Rủi ro về việc phân phối thẻ thông minh* Vấn đề cuối cùng mà các thẻ thông minh sẽ phải đối mặt trong việc khuếch tán rộng rãi liên quan đến việc bổ sung sản phẩm. Mặc dù các thẻ thông minh tương đối rẻ, nhưng đầu đọc thẻ thì không (khoảng 50 đến 200 đô la). Tuy nhiên, trong một nỗ lực để làm cho thẻ thông minh phổ biến hơn, các công ty như Netscape và Microsoft đang đề xuất đưa phần mềm vào các gói mà họ tạo ra. Nếu được sử dụng làm thẻ thanh toán, không phải mọi cửa hàng hoặc nhà hàng sẽ có phần cứng cần thiết để sử dụng các loại thẻ này, vì công nghệ này an toàn hơn, cũng đắt hơn để sản xuất và sử dụng. Do đó, một số cửa hàng có thể tính một khoản phí tối thiểu cơ bản để sử dụng thẻ thông minh để thanh toán hơn là tiền mặt.

## Thách thức trong việc phát triển ứng dụng thẻ thông minh

Phát triển ứng dụng thẻ thông minh theo truyền thống là một quá trình dài và khó. Mặc dù các thẻ được chuẩn hóa về kích thước, hình dạng, và giao thức giao tiếp, các hoạt động bên trong khác nhau giữa các nhà sản xuất. Hầu hết các công cụ phát triển thẻ thông minh được xây dựng bởi các nhà sản xuất thẻ thông minh bằng cách sử dụng các công cụ ngôn ngữ lắp ráp chung và giả lập phần cứng chuyên dụng thu được từ các nhà cung cấp chip silicon. Hầu như không thể cho các bên thứ ba phát triển các ứng dụng một cách độc lập và bán chúng cho các tổ chức phát hành. Do đó, việc phát triển các ứng dụng thẻ thông minh đã được giới hạn trong một nhóm các chuyên gia giàu kinh nghiệm và chuyên viên lập trình, những người có kiến thức sâu rộng về phần cứng và phần mềm thẻ thông minh cụ thể.

Vì không có giao diện ứng dụng tiêu chuẩn cao cấp sẵn có trong thẻ thông minh, nên các nhà phát triển ứng dụng cần phải xử lý các giao thức giao tiếp mức thấp, quản lý bộ nhớ và các chi tiết khác theo thời gian của phần cứng cụ thể của thẻ thông minh. Hầu hết các ứng dụng thẻ thông minh đang được sử dụng ngày nay đều được phát triển từ đầu, đó là một quá trình mất nhiều thời gian. Thường mất một hoặc hai năm cho một sản phẩm để đi vào thị trường. Việc nâng cấp phần mềm hoặc ứng dụng đến một nền tảng khác là đặc biệt khó khăn hoặc không thể.

Hơn nữa, các ứng dụng thẻ thông minh được phát triển để chạy trên nền tảng độc quyền, các ứng dụng từ các nhà cung cấp dịch vụ khác nhau không thể cùng tồn tại và cung cấp trên một thẻ duy nhất. Công nghệ Java Card là một giải pháp để vượt qua các trở ngại cản trở việc phát triển thẻ thông minh. Nó cho phép thẻ thông minh và thiết bị hạn chế bộ nhớ khác có thể chạy các ứng dụng (được gọi là applet) được viết bằng ngôn ngữ lập trình Java. Thông thường, công nghệ Java Card xác định nền tảng thẻ thông minh an toàn, di động và nhiều ứng dụng kết hợp nhiều lợi thế chính của ngôn ngữ Java .

## Các hình thức tấn công trên thẻ thông minh

Thẻ thông minh có khả năng bảo mật cao bởi các thành phần vật lý của con chip đều ở dạng siêu nhỏ và chúng đều có khả năng chống lại những tấn công vật lý. Tuy nhiên mạnh mẽ là thế các thẻ thông minh vẫn có những yếu điểm các hacker luôn tìm thấy cảm hứng từ các biến chip nhỏ bé, các hacker đã phát triển một loạt các kỹ thuật để quan sát và ngăn chặn các hoạt động của thẻ thông minh để có thể tước đoạt quyền truy cập thông tin, lấy các thông tin hữu ích cũng như chiếm đoạt thông tin đó. Dưới đây sẽ mô tả các cuộc tấn công trên thẻ thông minh, hiện nay có ba cuộc tấn công cơ bản: cuộc tấn công logic, cuộc tấn công phần cứng và cuộc tấn công kênh phụ (side - channel).

**Cuộc tấn công Logic**: dựa vào những suy luận logic liên quan đến các thuật toán mã hóa hacker cố gắng khai thác lỗ hổng trong các lĩnh vực sau:

* *Triển khai phần mềm*: Thẻ thông minh là một bộ vi xử lý, khi triển khai thẻ thông minh thì phải thực hiện các lệnh để chạy, chip hỗ trợ hàng ngàn lệnh bổ sung. Do đó chức năng này có thể bị lạm dụng cho việc thu thập dữ liệu không mong muốn cho mục đích sửa đổi.
* *Các lệnh ẩn*: Các hệ thống thẻ thông minh hỗ trợ trên 65000 lệnh trên lý thuyết, mặc dù chỉ có một vài lệnh trong số đó là cần thiết cho một ứng dụng cụ thể, phần còn lại dễ bị lạm dụng cho một mục đích khác.
* *Định vị thông số và tràn bộ đệm*: Nhập các giá trị tham số không hợp lệ, có thể không được phép hoặc vượt quá chiều dài, sẽ dẫn đến kết quả không mong muốn ví dụ như tràn bộ đệm.
* *Giao thức mã hoá, thiết kế và cài đặt*: Các giao thức điều khiển mật mã và các hoạt động mật mã trên thẻ thông minh. Nếu giao thức không được thiết kế cẩn thận, những sai sót ẩn này có thể ảnh hưởng đến hoạt động của chip. Ví dụ, một số thẻ có các phương pháp dự phòng để nâng cao độ tin cậy trong trường hợp các vấn đề kỹ thuật, trong khi điều này là không an toàn và kẻ tấn công có thể có lợi từ việc tạo ra các chức năng giả tưởng.

**Cuộc tấn công phần cứng**: đòi hỏi các thiết bị hiện đại để có thể xâm nhập vào các vi mạch của thẻ (chip), hacker khai thác lỗi trong các lĩnh vực sau:

* *Tấn công xâm nhập* (invasive attack): lớp bảo vệ của mạch sẽ bị phá bỏ, các kỹ thuật xử lý ảnh được sử dụng để quan sát các lớp và cấu trúc mạch, các tiếp xúc điện được kết nối và bộ nhớ của thẻ sẽ được đọc. Cách tấn công này đòi hỏi những thiết bị hiện đại, chính xác và đắt tiền thường chỉ có những phòng thì nghiệm cao cấp mới có được.
* *Tấn công nửa xâm nhập* (semi-invasive attack): khác với loại tấn công trên thì trong loại tấn công này các mạch điện không được kết nối. Một ví dụ điển hình là việc dùng tia laser chiếu vào mạch để mạch hoạt động không bình thường. Qua việc xử lý các kết quả không bình thường đó thì có thể dò được một vài thông tin bí mật.
* *Dung môi hóa học, chất tẩy, chất nhuộm*: Các chất này là các chất khử có thể phát hiện các khối cấu hình của chip từ đó nó có thể phân tích và khai thác thông tin từ chip.
* *Kính hiển vi quang học và điện tử*: Mặc dù kích thước của chip nhỏ hơn 1 micro, nhưng nó vẫn có thể nhìn thấy qua điện tử, ngay cả kính hiển vi quang học. Một chip được thiết kế cẩn thận vẫn có thể được phân tích để tiết lộ các phần hoạt động của nó, chạy mã và thậm chí cả giá trị trên bus dữ liệu
* *Trạm thăm dò:* một đầu dò nhỏ được đặt trên một đường dây tùy ý để tạo ra một kênh mới. Nếu bus dữ liệu có thể được định vị qua hai cách tiếp cận trên, tất cả các truyền dữ liệu có thể bị chặn, chẳng hạn như mã lập trình, dữ liệu lập trình, bao gồm các phím.

**Tấn công qua kênh phụ**: Một cuộc tấn công kênh phụ cố gắng khai thác một số hiện tượng vật lý để phân tích hoặc sửa đổi hành vi của thẻ thông minh, chẳng hạn như thời gian thực hiện thao tác, năng lượng tiêu thụ điện, cường độ của điện trường vv…

* *Thời gian thực hiện thao tác:* Thông qua việc phân tích thời gian mà thẻ thực hiện các thao tác (cộng, nhân, lũy thừa) ta có thể suy ra giá trị của một hay nhiều bit mà ta đang cần tìm. Tấn công đầu tiên sử dụng thông tin về thời gian thao tác là năm 1996 để tấn công các thuật toán RSA, DES và Difie-Hellman.
* *Năng lượng tiêu thụ điện:* Lượng điện tiêu thụ bởi thẻ vào mỗi thời điểm phụ thuộc các giá trị trung gian của thuật toán vào thời điểm đó, do đó ta có thể dùng các tín hiệu về năng lượng tiêu thụ điện để tấn công. Công bố kết quả bởi Paul Kocher năm 1999.
* *Cường độ của điện trường:* Do tồn tại một mối quan hệ giữa cường độ dòng điện và cường độ trường điện từ, vì vậy, nếu thông tin bí mật bị rò rỉ thông qua các tín hiệu về năng lượng tiêu thụ thì các thông tin này cũng sẽ bị rò rỉ thông quan các tín hiệu về trường điện từ.

# CHƯƠNG 2: CÔNG NGHỆ JAVACARD

Thẻ thông minh có nhiều ưu điểm nhưng cũng có thách thức để phát triển thẻ thông minh. Các ứng dụng thẻ thông minh được phát triển để chạy trên nền tảng độc quyền, các ứng dụng từ các nhà cung cấp dịch vụ khác nhau không thể cùng tồn tại và cung cấp trên một thẻ duy nhất. Đó là trở ngại cản trở việc phát triển ứng dụng trên thẻ thông minh. Công nghệ JavaCard được hình thành nhằm giải quyết vấn đề trên. Chương 2 đi sâu vào nền tảng công nghệ JavaCard.

## Giới thiệu JavaCard

JavaCard là một công nghệ cho phép mang đến cho các trình ứng dụng Java applet có thể hoạt động một cách an toàn và bảo mật trên thẻ thông minh tương tự với các bộ nhớ nhỏ của các thiết bị lưu vết. Nó là nền tảng Java nhỏ nhất hướng tới các thiết bị nhúng. JavaCard là một phần nhỏ của Java được phát triển bởi Sun, được tích hợp bên trong các thiết bị, nó đơn giản hoá việc lập trình thẻ thông minh vì các tính năng hướng đối tượng của nó. JavaCard mang đến cho người dùng khả năng lập trình cho các thiết bị mà tạo các trình ứng dụng chuyên biệt. Nó được sử dụng rộng rãi trong SIM card (trong GSM của điện thoại di động) và thẻ ATM.

Ngôn ngữ lập trình Java có nhiều ưu điểm như tính đơn giản, tính di động, mô hình bảo mật và tính hướng đối tượng. Đó cũng chính là đặc tính mà JavaCard hướng tới là khả năng tương thích và bảo mật. Tính tương thích được thể hiện ở việc JavaCard nhắm tới mục tiêu tạo ra một môi trường chuẩn cho thẻ thông minh, cho phép các ứng dụng Java có thể chạy trên các loại thẻ thông minh khác nhau. Tương tự như với môi trường Java, JavaCard thực hiện bằng việc kết hợp một máy ảo JavaCard với một bộ thư viện chung. Tuy nhiên tính tương thích bị ảnh hưởng lớn bởi sự khác nhau về kích thước bộ nhớ, khả năng xử lý và hỗ trợ của các loại thẻ khác nhau.

JavaCard có một số đặc điểm khác với Java thông thường. Về mặt ngôn ngữ lập trình, JavaCard là một phần thu gọn của Java, các cú pháp của Java, giống như tất cả các biến thể của Java và làm cho lập trình viên viết mã dễ dàng hơn vì không cần phải học một cú pháp nào khác. Tuy nhiên, cú pháp quen thuộc không có nghĩa là dễ, thật dễ dàng để bị một quan niệm sai lầm trong suy nghĩ rằng tiêu chuẩn Java và Java Card rất giống nhau, điều này có thể xảy ra do cùng cú pháp, tên tương tự và toán tử, mô hình đối tượng, nhưng trên thực tế, hai môi trường này là tương đối khác nhau từ quan điểm lập trình.

JavaCard phát triển với mục đích lưu trữ các thông tin nhạy cảm, công nghệ Java Card luôn đề cao tính bảo mật và đảm bảo điều này bằng các yếu tố khác nhau như: đóng gói dữ liệu, tường lửa ngăn cách ứng dụng, mã hóa dữ liệu, tạo ứng dụng dạng Applet. Khả năng đóng gói dữ liệu cho phép dữ liệu được lưu trữ bên trong ứng dụng và các ứng dụng này được thực thi trên một máy ảo tách biệt với hệ điều hành và phần cứng của thẻ. Mỗi ứng dụng khác nhau lưu trữ trên JavaCard đều được ngăn cách bởi một tường lửa để hạn chế và kiểm tra được sự truy cập dữ liệu từ ứng dụng này sang ứng dụng khác. Khả năng mã hóa của JavaCard cho phép dữ liệu được mã hóa bằng các dạng mã hóa thông dụng sử dụng khóa như mã hóa DES, 3DES, AES hay RSA.

JavaCard là một thẻ thông minh có thể thực thi mã bytecode tương tự như Java. Do tiêu chuẩn Java quá lớn để phù hợp với thẻ thông minh, nên giải pháp cho vấn đề này là tạo ra một JavaCard với một số chuẩn Java đã được loại bỏ. JavaCard được hình thành dựa trên một tập con của Java API cộng với một số lệnh thẻ đặc biệt. Bên cạnh việc cung cấp cho các nhà phát triển môi trường phát triển quen thuộc, JavaCard còn cho phép các thẻ thông minh có thể có nhiều ứng dụng trên đó. Hầu hết các sản phẩm thẻ thông minh hiện có chỉ có một ứng dụng trên mỗi thẻ. Ứng dụng này được tự động gọi ra khi điện được cung cấp cho thẻ hoặc thẻ được cài đặt lại. Thẻ Java cho phép nhiều ứng dụng, có khả năng được viết bởi các tổ chức khác nhau, để tồn tại trên cùng một thẻ.

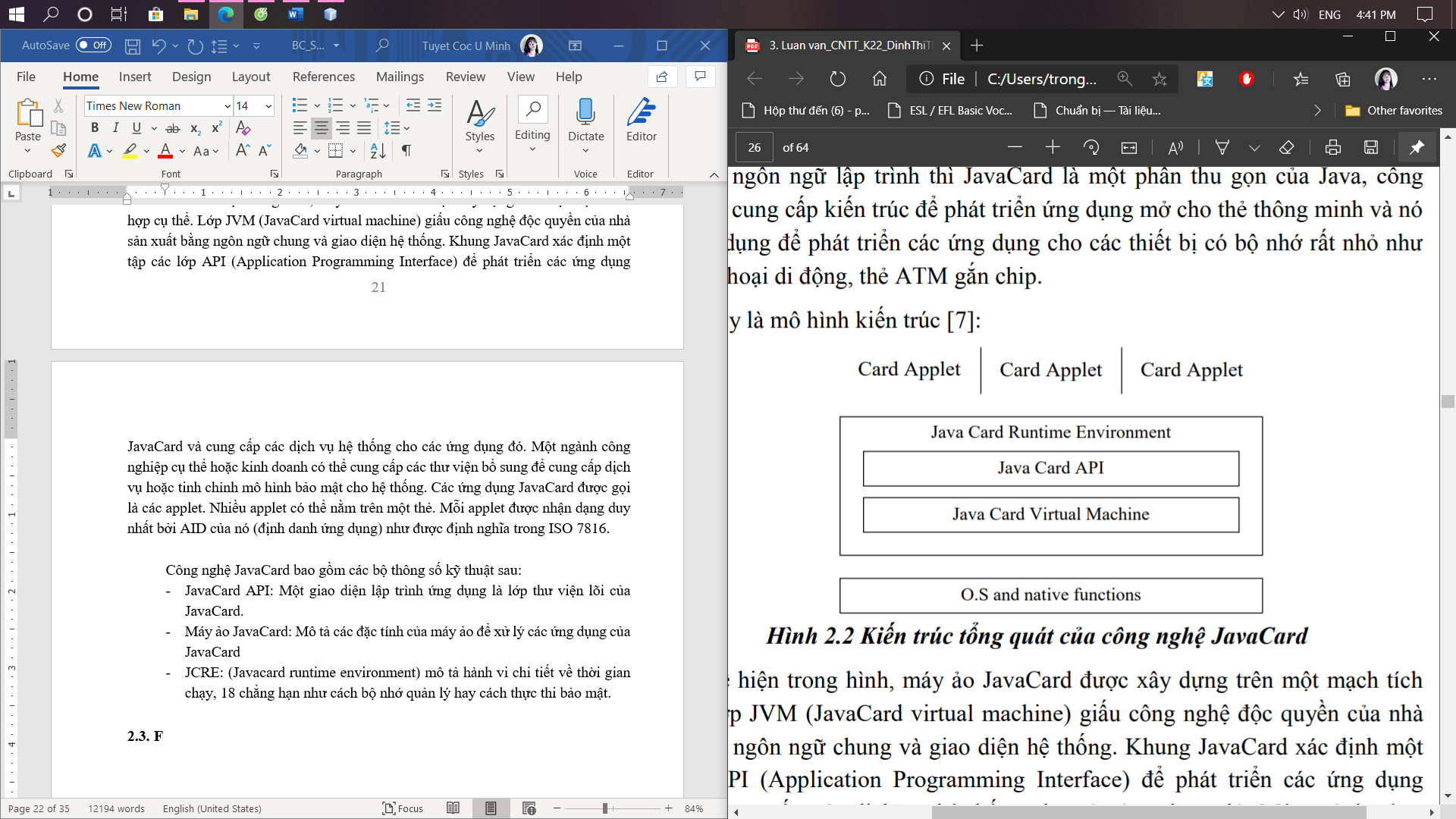
Dưới đây là những lợi ích chính của JavaCard cũng như ưu điểm của Sun:

* Đa nền tảng: JCRE (JavaCard Runtime Environment – môi trường chạy java card) phù hợp với tất cả các loại bộ xử lý thẻ thông minh.
* Tương thích: Các applet JavaCard được phát triển bởi một nhà cung cấp sẽ chạy trên bất kỳ thẻ nào phù hợp với đặc tả của JavaCard.
* Thẻ Java tuân thủ các tiêu chuẩn thẻ thông minh hiện có.
* Hỗ trợ tất cả các loại ứng dụng thẻ thông minh.

## Kiến trúc JavaCardfe

Về mặt ngôn ngữ lập trình thì JavaCard là một phần thu gọn của Java, công nghệ JavaCard cung cấp kiến trúc để phát triển ứng dụng mở cho thẻ thông minh và nó cũng được sử dụng để phát triển các ứng dụng cho các thiết bị có bộ nhớ rất nhỏ như SIM cho điện thoại di động, thẻ ATM gắn chip

Như thể hiện trong hình, máy ảo JavaCard được xây dựng trên một mạch tích hợp cụ thể. Lớp JVM (JavaCard virtual machine) giấu công nghệ độc quyền của nhà sản xuất bằng ngôn ngữ chung và giao diện hệ thống. Khung JavaCard xác định một tập các lớp API (Application Programming Interface) để phát triển các ứng dụng JavaCard và cung cấp các dịch vụ hệ thống cho các ứng dụng đó. Một ngành công nghiệp cụ thể hoặc kinh doanh có thể cung cấp các thư viện bổ sung để cung cấp dịch vụ hoặc tinh chỉnh mô hình bảo mật cho hệ thống. Các ứng dụng JavaCard được gọi là các applet. Nhiều applet có thể nằm trên một thẻ. Mỗi applet được nhận dạng duy nhất bởi AID của nó (định danh ứng dụng) như được định nghĩa trong ISO 7816.



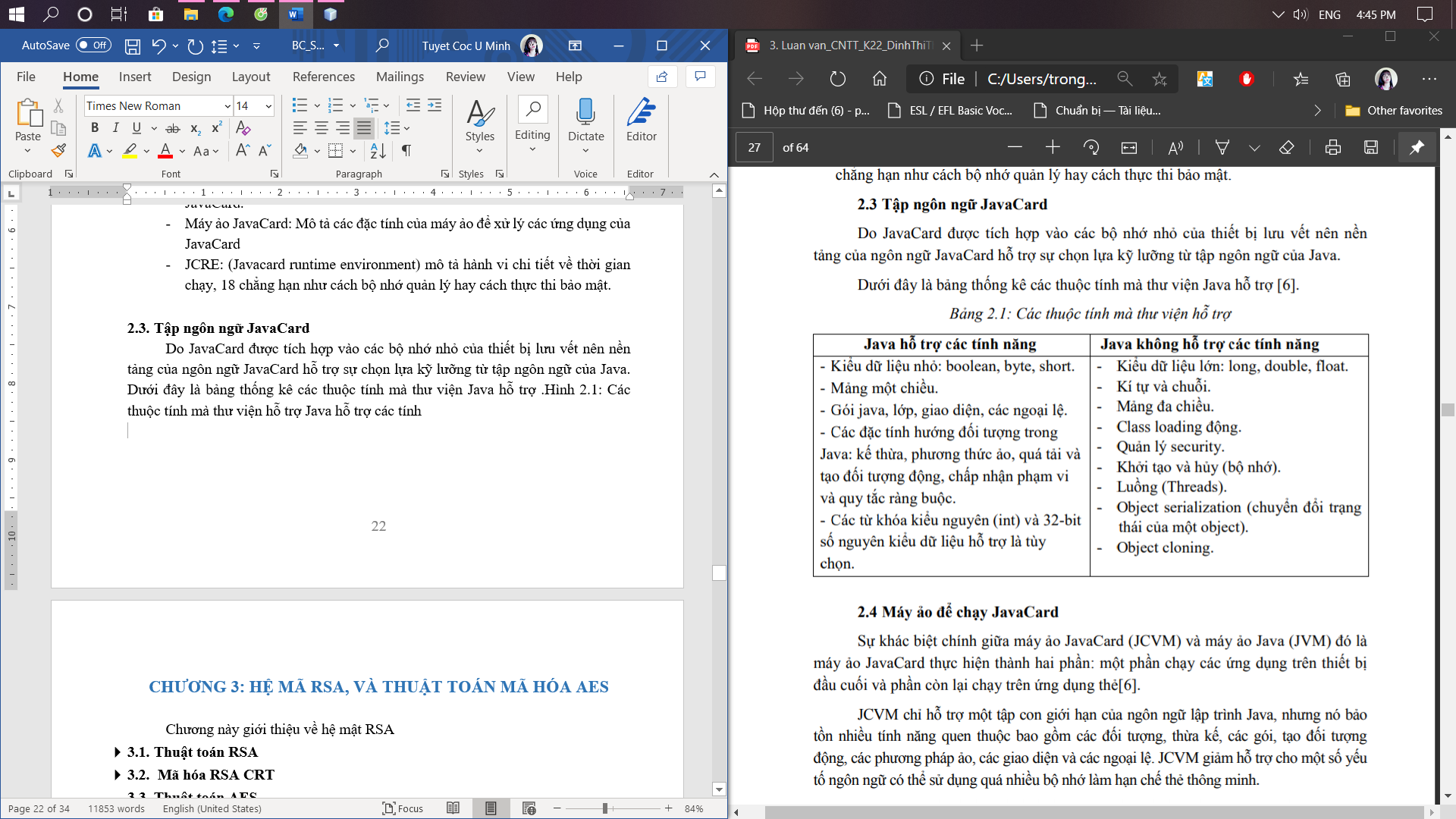
Hình 2.1 Kiến trúc tổng quát của công nghệ Javacard

Công nghệ JavaCard bao gồm các bộ thông số kỹ thuật sau:

* JavaCard API: Một giao diện lập trình ứng dụng là lớp thư viện lõi của JavaCard.
* Máy ảo JavaCard: Mô tả các đặc tính của máy ảo để xử lý các ứng dụng của JavaCard
* JCRE: (Javacard runtime environment) mô tả hành vi chi tiết về thời gian chạy, 18 chẳng hạn như cách bộ nhớ quản lý hay cách thực thi bảo mật.

## Tập ngôn ngữ JavaCard

Do JavaCard được tích hợp vào các bộ nhớ nhỏ của thiết bị lưu vết nên nền tảng của ngôn ngữ JavaCard hỗ trợ sự chọn lựa kỹ lưỡng từ tập ngôn ngữ của Java. Dưới đây là bảng thống kê các thuộc tính mà thư viện Java hỗ trợ .Hình 2.2: Các thuộc tính mà thư viện hỗ trợ Java hỗ trợ.



Hình 2.2 Các thuộc tính mà thư viện hỗ trợ

## Máy ảo để chạy JavaCard

Sự khác biệt chính giữa máy ảo JavaCard (JCVM) và máy ảo Java (JVM) đó là máy ảo JavaCard thực hiện thành hai phần: một phần chạy các ứng dụng trên thiết bị đầu cuối và phần còn lại chạy trên ứng dụng thẻ.

JCVM chỉ hỗ trợ một tập con giới hạn của ngôn ngữ lập trình Java, nhưng nó bảo tồn nhiều tính năng quen thuộc bao gồm các đối tượng, thừa kế, các gói, tạo đối tượng động, các phương pháp ảo, các giao diện và các ngoại lệ. JCVM giảm hỗ trợ cho một số yếu tố ngôn ngữ có thể sử dụng quá nhiều bộ nhớ làm hạn chế thẻ thông minh. Hình 2.3 mô tả máy ảo của javacard.



Hình 2.3 Máy ảo của javacard.

Máy ảo trên thẻ (on – card) interpreter (hay còn gọi là trình thông dịch JavaCard) cung cấp hỗ trợ thời gian chạy mô hình ngôn ngữ Java và cho phép độc lập phần cứng thực hiện các nhiệm vụ:

* Thực thi các đoạn mã bytecode và cuối cùng thực thi applet.
* Kiểm soát việc cấp phát bộ nhớ và tạo đối tượng.
* Đóng vai trò quan trọng trong việc đảm bảo an ninh thời gian chạy như ngăn cách tường lửa giữa các ứng dụng và hỗ trợ chia sẻ dữ liệu một cách bảo mật.

Phần máy ảo Java trên các thiết bị đầu cuối (off – card), nó chứa công cụ JCC (javacard converter) hỗ trợ việc xác thực, đóng gói, tối ưu mã hóa thường được gọi là công cụ chuyển đổi thẻ Java. Đầu ra của công cụ chuyển đổi là một tệp tin Applet được Chuyển đổi (CAP), tệp chứa tất cả các lớp trong một gói Java trong một biểu diễn nhị phân có thể chạy được. Các lớp động không được hỗ trợ trong Java Card bởi:

* Các lớp động cần phải truy cập vào những vị trí lưu trữ của tệp mô tả lớp, điều này là không khả dụng trong môi trường thẻ.
* Vấn đề an ninh (bảo mật) trong thẻ thông minh nghiêm cấm các lớp động (đối tượng ràng buộc động được cho phép).
* Thẻ thông minh hạn chế về tài nguyên, bộ nhớ. Bộ chuyển đổi đóng gói tất cả các lớp được ứng dụng tham chiếu vào một gói ứng dụng. Lúc này bộ công cụ đóng vai trò là một bộ tiền xử lý gói ứng dụng JavaCard với các bước như sau:
* Xác thực: Kiểm tra xem gói ứng dụng đã được định dạng đúng với đầy đủ các bảng ký hiệu và kiểm tra các phạm vi ngôn ngữ lập trình có tuân thủ các đặc tả của Java Card hay không.
* Chuẩn bị: Phân bổ lưu trữ để tạo ra kiến trúc dữ liệu cho máy ảo tương ứng với các lớp, tạo các trường và phương thức tĩnh và khởi tạo các biến tĩnh thành các giá trị mặc định.
* Quyết định: Thay thế các tham chiếu ký tự tới phương thức và biến bằng tham chiếu trực tiếp khi có thể.

Thực hiện ba bước này trong bộ công cụ chuyển đổi trước khi một applet được cài lên thẻ cho phép máy ảo trên thẻ trở nên nhỏ gọn và hiệu quả hơn. Một khi ứng dụng đã được cài lên thẻ thông minh nó được coi là nạp và sẵn sàng để chạy (mặc dù một số khởi tạo có thể được yêu cầu). Sau đó, JCRE (JavaCard Run Environment) sẽ thực hiện một số thủ tục khởi tạo để tạo các tham số được khai báo với các giá trị mặc định trong giao diện. Mặc dù JCC (JavaCard converter) thực hiện càng sớm càng bắt buộc và giải quyết càng tốt, một số ràng buộc cuối cũng được hỗ trợ bởi môi trường chạy JavaCard.

Cap File: chứa một biểu diễn nhị phân thực thi các lớp trong một package Java. Một tệp CAP là một tệp JAR – là định dạng chứa tệp CAP, nó chứa một bộ thành phần được lưu trữ dưới dạng tệp cá nhân trong tệp JAR, tệp JAR chia thành các thành phần nhỏ trong mỗi thành phần nhỏ chứa nội dung của tệp CAP như thông tin về lớp, thực thi bytecode, liên kết thông tin, xác minh thông tin….Định dạng tệp CAP cho phép tối ưu hóa bằng cách sử dụng cấu trúc dữ liệu phù hợp và hạn chế gián tiếp. Nó định nghĩa một tập lệnh bytecode dựa trên tối ưu hóa từ tập lệnh Java bytecode (Mã bytecode là mã trung gian, chưa phải là mã máy).

Java được tạo ra với tiêu chí "viết (code) một lần, thực thi khắp nơi" ("Write Once, Run Anywhere" (WORA)). Chương trình phần mềm viết bằng Java có thể chạy trên mọi nền tảng (platform) khác nhau thông qua một môi trường thực thi với điều kiện có môi trường thực thi thích hợp hỗ trợ nền tảng đó. Trong công nghệ Java, tệp lớp là phần trung tâm của kiến trúc Java, nó định nghĩa tiêu chuẩn cho sự tương thích nhị phân của nền tảng Java. Do đặc tính phân bố của kiến trúc hệ thống JavaCard, tệp CAP đặt định dạng tệp chuẩn cho sự tương thích nhị phân trên nền tảng thẻ Java. Định dạng tệp CAP là hình thức trong đó phần mềm được tải lên thẻ thông minh Java. Ví dụ, các miếng CAP cho phép tải động các lớp applet sau khi thẻ đã được thực hiện. Đó là cách nó được lấy tên tệp chuyển đổi tên (CAP).

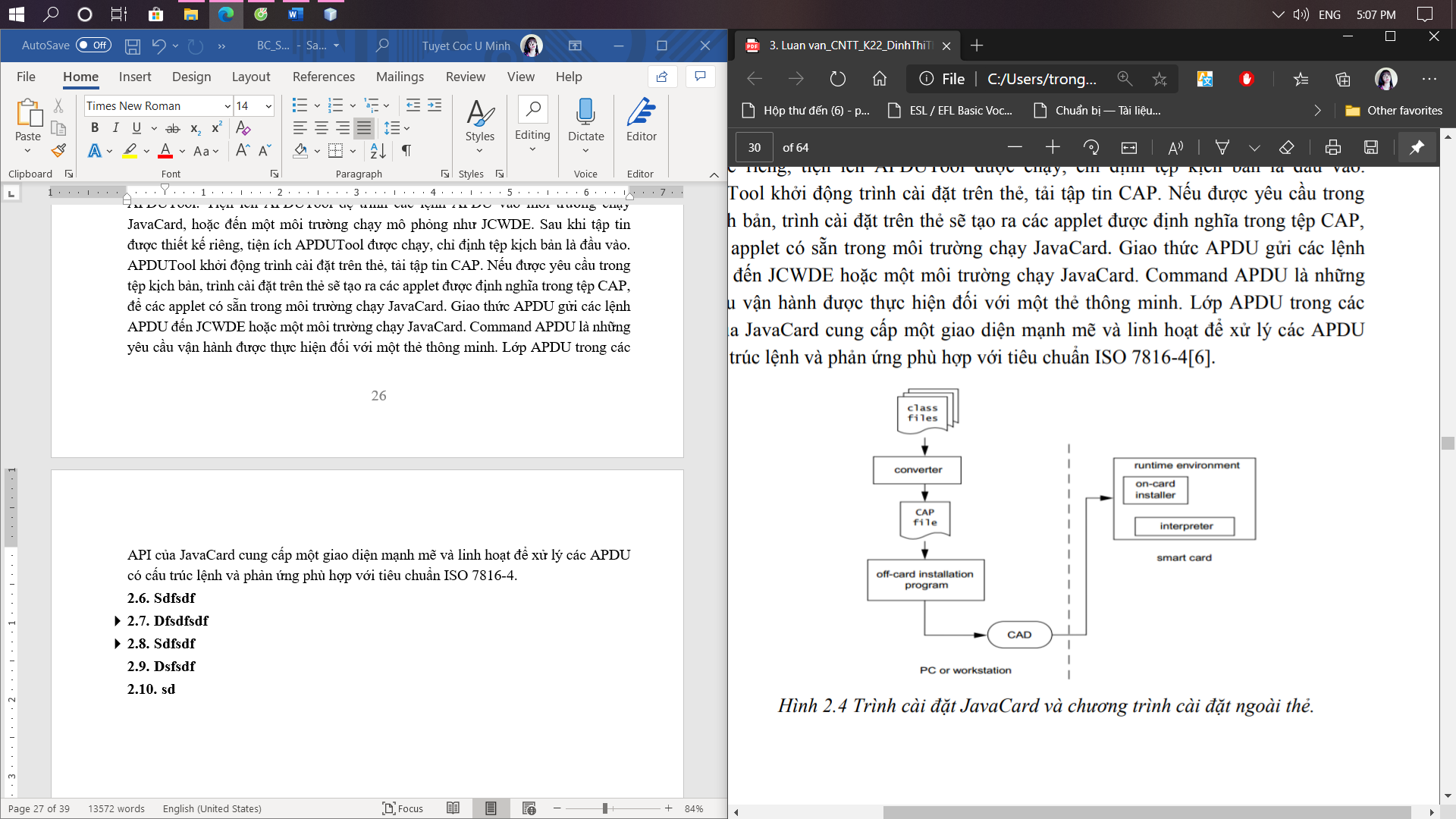
## Cài đặt JavaCard và chương trình cài đặt trên thiết bị (Off-Card)

Các công cụ sau đây yêu cầu cài đặt ứng dụng JavaCard:

* Một công cụ chuyển đổi để chuyển đổi một applet JavaCard sang một định dạng cần thiết để cài đặt.
* Các công cụ xác minh ngoài thẻ để kiểm tra tính toàn vẹn của các tệp được tạo ra bởi Converter.
* Trình cài đặt thẻ gắn ngoài để cài đặt một thẻ JavaCard vào một thẻ thông minh các applet JavaCard được phát triển bằng cách sử dụng các lớp và các công cụ trên máy trạm hoặc máy tính cá nhân. Cụ thể nó cho phép nhà phát triển:
  + Biên dịch applet.
  + Theo tùy chọn, kiểm tra applet trong JCWDE (JavaCard Workstation Development Environment), và gỡ lỗi applet này. JCWDE, chạy trên máy trạm hoặc máy tính cá nhân, mô phỏng môi trường chạy JavaCard trên một máy ảo Java. Nó không phải là một mô phỏng hoàn chỉnh, ví dụ JCWDE không mô phỏng tường lửa applet của một JCVM. Tuy nhiên, JCWDE cung cấp mô phỏng cho phép kiểm tra ban đầu về một thẻ JavaCard applet. Nó cho phép chạy một applet như thể nó đã được che đậy trong bộ nhớ và đọc của một thẻ thông minh. Và quan trọng hơn, nó cho phép chạy thử nghiệm trong một máy trạm hoặc máy PC, mà không cần phải chuyển đổi các applet, tạo ra một tập tin “mask”, hoặc cài đặt các applet.
  + Chuyển đổi các applet và tất cả các lớp để cài đặt vào một tập tin CAP, và có thể là một tập tin được “export”. Một tập tin export được sử dụng để chuyển đổi một gói khác nếu các gói import các class từ gói này. Không giống như JVM, xử lý một lớp cùng một lúc, đơn vị chuyển đổi của bộ chuyển đổi là một gói. Trình biên dịch Java tạo ra các tệp class từ mã nguồn, sau đó trình chuyển đổi xử lý tất cả các tệp lớp tạo thành gói Java và chuyển gói đó sang tệp CAP.

Trong công nghệ Java Card, một applet JavaCard không trực tiếp kết hợp vào một mask (mặt nạ). Tương tự như vậy, sau khi sản xuất một thẻ thông minh, một applet JavaCard không trực tiếp tải về để cài đặt vào một thẻ thông minh. Thay vào đó, đối với lớp mask (mặt nạ), một lớp applet và tất cả các lớp trong gói của nó chuyển đổi sang một tệp applet JavaCard (JCA). Tệp JCA cho bất kỳ gói nào khác được bao gồm trong mask sau đó sẽ được chuyển đổi thành định dạng tương thích với môi trường chạy đích. Đây là kết quả được chuyển đổi cho môi trường chạy đích mà được kết hợp vào mask. Cả tệp JCA và tệp CAP đều là tệp tự mô tả. Các tệp này chứa thông tin về gói được chuyển đổi.

Như đã đề cập trước đó, một applet JavaCard không cài đặt vào một thẻ thông minh, thay vì nó đã được cài đặt tệp CAP. Trình cài đặt thẻ ra tạo một tệp kịch bản chứa các APDU lệnh xác định phần đầu và kết thúc của tệp CAP, các thành phần và dữ liệu thành phần. Tập tin kịch bản được sử dụng làm đầu vào cho tiện ích APDUTool. Tiện ích APDUTool đệ trình các lệnh APDU vào môi trường chạy JavaCard, hoặc đến một môi trường chạy mô phỏng như JCWDE. Sau khi tập tin được thiết kế riêng, tiện ích APDUTool được chạy, chỉ định tệp kịch bản là đầu vào. APDUTool khởi động trình cài đặt trên thẻ, tải tập tin CAP. Nếu được yêu cầu trong tệp kịch bản, trình cài đặt trên thẻ sẽ tạo ra các applet được định nghĩa trong tệp CAP, để các applet có sẵn trong môi trường chạy JavaCard. Giao thức APDU gửi các lệnh APDU đến JCWDE hoặc một môi trường chạy JavaCard. Command APDU là những yêu cầu vận hành được thực hiện đối với một thẻ thông minh. Lớp APDU trong các API của JavaCard cung cấp một giao diện mạnh mẽ và linh hoạt để xử lý các APDU có cấu trúc lệnh và phản ứng phù hợp với tiêu chuẩn ISO 7816-4.



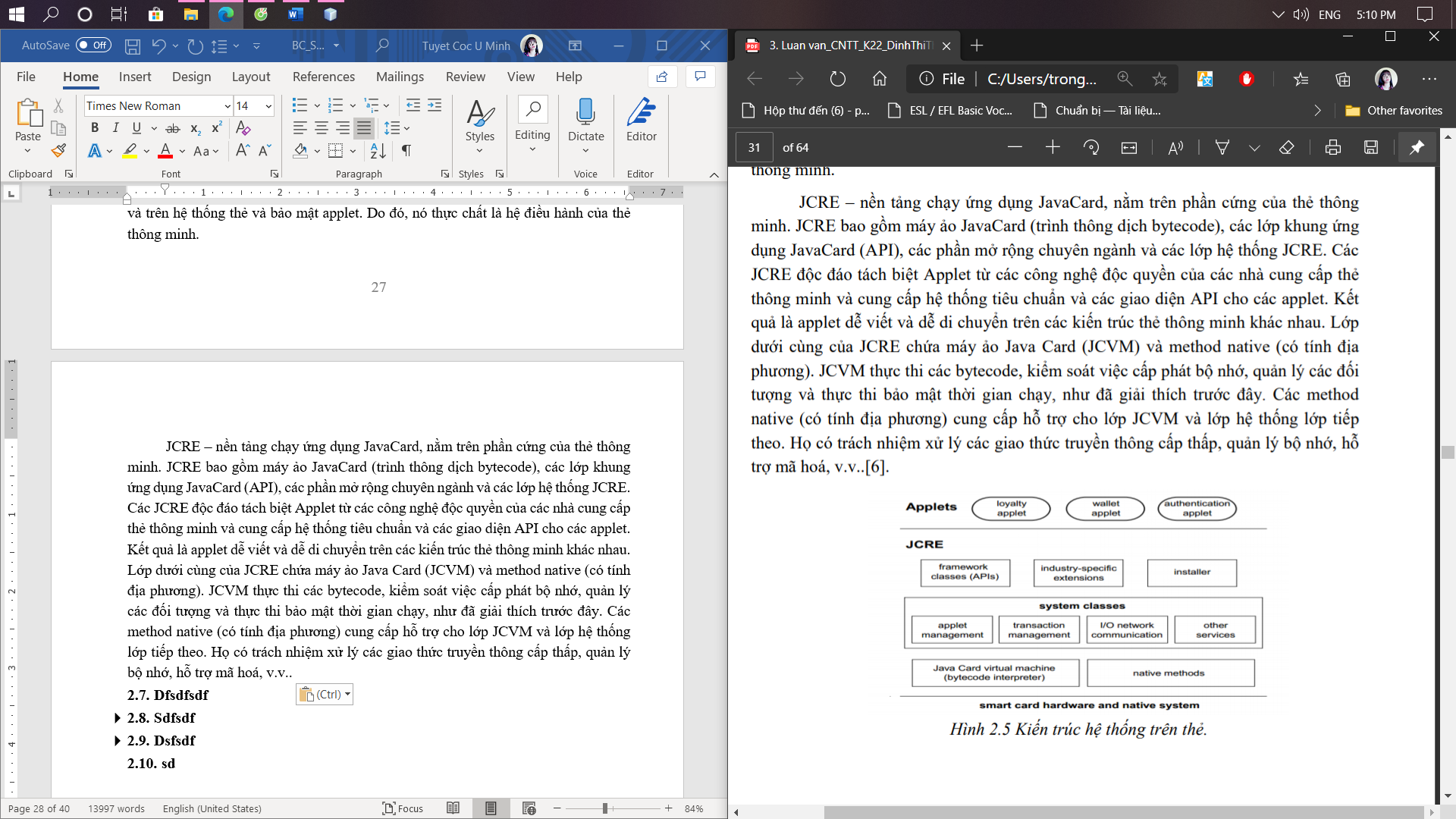
Hình 2.4 Trình cài đặt JavaCard và chương trình cài đặt ngoài thẻ

Các lệnh APDU luôn luôn là các cặp, mỗi cặp có chứa một lệnh APDU, nó chỉ định lệnh, và một APDU phản hồi, nó sẽ gửi lại kết quả thi hành của lệnh. Trong thế giới thẻ, thẻ thông minh là những người giao tiếp phản hồi - nghĩa là họ không bao giờ bắt đầu truyền thông, họ chỉ đáp ứng với APDU từ thế giới bên ngoài. Ứng dụng đầu cuối gửi một lệnh APDU qua CAD. JCRE nhận lệnh, và chọn một applet mới hoặc chuyển lệnh tới applet hiện đang được chọn. Applet hiện đang được lựa chọn xử lý lệnh và trả về một APDU phản ứng cho ứng dụng đầu cuối. Command APDU và APDU phản hồi được trao đổi luân phiên giữa thẻ và CAD.

## Môi trường chạy JavaCard

Trình cài đặt và trình biên dịch không phải là thành phần hệ thống duy nhất chạy trên JavaCard, nó còn nhiều các thành phần khác. Môi trường chạy JavaCard (JCRE) bao gồm các thành phần hệ thống thẻ Java chạy bên trong một thẻ thông minh. JCRE chịu trách nhiệm về quản lý tài nguyên thẻ, truyền thông mạng, thực hiện applet và trên hệ thống thẻ và bảo mật applet. Do đó, nó thực chất là hệ điều hành của thẻ thông minh.

JCRE – nền tảng chạy ứng dụng JavaCard, nằm trên phần cứng của thẻ thông minh. JCRE bao gồm máy ảo JavaCard (trình thông dịch bytecode), các lớp khung ứng dụng JavaCard (API), các phần mở rộng chuyên ngành và các lớp hệ thống JCRE. Các JCRE độc đáo tách biệt Applet từ các công nghệ độc quyền của các nhà cung cấp thẻ thông minh và cung cấp hệ thống tiêu chuẩn và các giao diện API cho các applet. Kết quả là applet dễ viết và dễ di chuyển trên các kiến trúc thẻ thông minh khác nhau. Lớp dưới cùng của JCRE chứa máy ảo Java Card (JCVM) và method native (có tính địa phương). JCVM thực thi các bytecode, kiểm soát việc cấp phát bộ nhớ, quản lý các đối tượng và thực thi bảo mật thời gian chạy, như đã giải thích trước đây. Các method native (có tính địa phương) cung cấp hỗ trợ cho lớp JCVM và lớp hệ thống lớp tiếp theo. Họ có trách nhiệm xử lý các giao thức truyền thông cấp thấp, quản lý bộ nhớ, hỗ trợ mã hoá, v.v..



Hình 2.5 Kiến trúc hệ thống trên thẻ

JCVM xử lý việc cấp phát bộ nhớ, thực hiện bytecode, quản lý đối tượng và bảo mật. Các method native (có tính địa phương) xử lý giao thức truyền thông mức thấp, quản lý bộ nhớ, hỗ trợ mã hoá và còn nhiều hơn nữa. Các lớp Hệ thống xử lý các nhiệm vụ mà một lõi hệ điều hành bình thường sẽ và gọi các method native (có tính địa phương) để thực hiện việc này. Các lớp API là các thư viện nhỏ gọn làm cho việc tạo các applet trở nên dễ dàng hơn. Trình cài đặt nạp applet vào thẻ, các phần mở rộng khác là các thư viện chuyên ngành, ví dụ như Open Platform mở rộng các dịch vụ JCRE để đáp ứng nhu cầu bảo mật đặc biệt của các ngân hàng.

JCRE được nạp vào thẻ Java tại nhà máy và vẫn duy trì ở đó cho đến khi thẻ bị phá hủy. Khi thẻ được đặt trong CAD (Card Accepting Device), nó được kích hoạt và bắt đầu sao chép dữ liệu từ chương trình từ EEPROM và ROM sang RAM nhanh hơn. Trong quá trình giao dịch, dữ liệu và các đối tượng phải được bảo toàn và được sao chép từ RAM vào EEPROM. EEPROM giữ dữ liệu khi không có điện, khi điện bị mất thẻ sẽ chuyển sang ngủ đông.

## API JavaCard

Vì các ứng dụng JavaCard chủ yếu liên quan đến nhiều công ty phát hành thẻ, tính tương thích phải được thiết kế ngay từ đầu. Từ quan điểm kỹ thuật, chìa khóa là một JavaCard API đây là một lớp phần mềm cho phép ứng dụng giao tiếp với thẻ thông minh và đầu đọc của nhiều nhà sản xuất.

API có thể được coi là một thiết bị chuyên dụng hoạt động như một lớp phiên dịch giữa một ứng dụng và thẻ. API cho phép chạy ứng dụng để chọn thẻ thông minh từ nhiều nhà cung cấp. Việc mở một ứng dụng cho nhiều thẻ Java khuyến khích cạnh tranh giữa các nhà cung cấp thẻ và lợi ích của sự cạnh tranh đó - chất lượng tốt hơn và giá thấp hơn.

Một API không phải là một giao diện phổ quát mà sẽ làm việc với tất cả các thẻ Java. Thay vào đó, nó cung cấp một cách để các ứng dụng gửi lệnh tới hệ điều hành chip cụ thể của nhiều thẻ. Các lập trình viên có thể bắt đầu bằng cách phát triển một API cho hai hoặc ba thẻ và theo thời gian, mở rộng phần mềm để bao gồm một hay nhiều JavaCard. Có một giới hạn thực tế về kích thước của chương trình phần mềm có thể được lưu trữ trong một số thiết bị đầu cuối di động, nhưng API nên đủ linh hoạt để chứa các thẻ từ các nhà cung cấp cạnh tranh API cũng có thể được sử dụng để kiểm soát các phiên bản dữ liệu. Nếu cần thay đổi các yếu tố dữ liệu trên thẻ sau khi phát hành, API có thể được sử dụng để cập nhật thẻ mà không cần phải nhớ lại thẻ để định dạng lại.

JavaCard API bao gồm một tập lớp cho ứng dụng thẻ thông minh theo mô hình ISO 7816. API bao gồm ba package lõi và một package mở rộng. Ba package lõi đó là java.lang, java.framework và java.security. Package mở rộng đó là java-cardx.crypto. Có nhiều lớp nền tảng Java không hỗ trợ trong JavaCard API, ví dụ trong Java có một số class về GUI interface, network I/O và hệ thống file I/O màn hình là không được hỗ trợ trong JavaCard. Đó là lý do tại sao thẻ thông minh không có hiển thị nó sử dụng giao thức mạng và cấu trúc file hệ thống.

Cụ thể hơn về các gói:

* Gói java.lang

Gói java.lang của JavaCard là một tập hợp con nghiêm ngặt của gói java.lang tương ứng trên nền tảng Java. Các lớp được hỗ trợ là Object, Throwable và một số lớp ngoại lệ liên quan đến máy ảo, như được trình bày trong bảng dưới. Đối với các class được hỗ trợ như đối tượng JavaCard Objectclass chỉ định nghĩa một constructor mặc định. Java.lang package cung cấp hỗ trợ ngôn ngữ Java cơ bản. Lớp Objectdefines một gốc cho phân cấp lớp JavaCard, và lớp Throwable cung cấp chung cho tất cả các ngoại lệ. Các lớp ngoại lệ được hỗ trợ đảm bảo ngữ nghĩa thống nhất khi một lỗi xảy ra do vi phạm ngôn ngữ Java. Ví dụ, cả máy ảo Java và máy ảo JavaCard thông qua NullPointerException một khi một tài liệu tham khảo null được truy cập.

Bảng các ngoại lệ:

|  |  |  |
| --- | --- | --- |
| Object | Exception | Throwable |
| RuntimeException | ArrayIndexOutOfBoundsException | ArithmeticException |
| ArrayStoreException | IndexOutOfBoundsException | ClassCastException |
| NullPointerException | NegativeArraySizeException | SecurityException |

* Gói javacard.framework

Các javacard.framework là một gói thiết yếu. Nó cung cấp các lớp khung và các giao diện cho các chức năng cốt lõi của một applet JavaCard. Quan trọng nhất, nó định nghĩa một lớp cơ sở Applet, cung cấp một khuôn khổ để thực hiện và tương tác với applet với JCRE trong suốt vòng đời applet. Vai trò của nó đối với JCRE cũng tương tự như của lớp Java Applet tới một trình duyệt lưu trữ. Một lớp applet người dùng phải mở rộng từ lớp cơ sở Applet và ghi đè các phương thức trong lớp Applet để thực hiện chức năng của applet.

Một lớp quan trọng khác trong gói javacard.framework là lớp APDU. APDU được thực hiện bởi các giao thức truyền dẫn. Hai giao thức truyền dẫn chuẩn là T = 0 và T = 1. APDU được thiết kế là giao thức truyền dẫn độc lập. Nói cách khác, nó được thiết kế cẩn thận sao cho những phức tạp và khác biệt giữa các giao thức T = 0 và T = 1 bị ẩn đi từ các nhà phát triển applet. Các nhà phát triển Applet có thể xử lý các lệnh của APDU dễ dàng hơn bằng cách sử dụng các phương pháp được cung cấp trong lớp APDU. Applet hoạt động chính xác bất kể giao thức truyền tải cơ bản mà nền tảng hỗ trợ.

Các nền tảng có gói java.lang.System là không được hỗ trợ. Nền tảng JavaCard cung cấp lớp javacard.framework.JCSystem, cung cấp một giao diện cho hành vi của hệ thống. JCSystemclass bao gồm một bộ sưu tập các phương pháp để kiểm soát việc thực hiện aplelet, quản lý tài nguyên, quản lý giao dịch, và chia sẻ đối tượng interapplet trên nền Java Card.

Các lớp khác được hỗ trợ trong gói javacard.frameworkpackage là mã PIN, tiện ích và trường hợp ngoại lệ. Mã PIN là viết tắt của số nhận dạng cá nhân. Đây là hình thức mật khẩu phổ biến nhất được sử dụng trong thẻ thông minh để xác thực chủ thẻ.

* Gói javacard.security

Gói Javacard.security cung cấp một khuôn khổ cho các chức năng mật mã được hỗ trợ trên nền JavaCard. Thiết kế của nó dựa trên gói java.security. Gói Javacard.security định nghĩa một khoá chính của nhà máy sản xuất các giao diện khác nhau đại diện cho các khóa mật mã được sử dụng trong các thuật toán đối xứng (DES) hoặc bất đối xứng (DSA và RSA). Ngoài ra, nó hỗ trợ các lớp cơ sở trừu tượng RandomData, Chữ ký và MessageDigest, được sử dụng để tạo ra dữ liệu ngẫu nhiên và để tính toán thông báo thư và chữ ký.

* Gói javacardx.crypto

Javacardx.crypto là một gói phần mở rộng. Nó chứa các lớp và giao diện mật mã phải tuân theo các yêu cầu về quản lý xuất khẩu của Hoa Kỳ. Gói Javacardx.crypto định nghĩa lớp cơ sở trừu tượng cho hỗ trợ chức năng mã hóa và giải mã.

Các gói javacard.security và gói javacardx.crypto xác định các giao diện API mà applet gọi để yêu cầu các dịch vụ mật mã. Tuy nhiên, họ không cung cấp bất kỳ thực hiện. Nhà cung cấp JCRE cần cung cấp các lớp triển khai các giao diện chính và mở rộng từ các lớp trừu tượng RandomData, Signature, MessageDigest và Cipher. Thông thường, một bộ xử lý đồng bộ riêng biệt tồn tại trên thẻ thông minh để thực hiện tính toán mật mã.

## Package và quy ước đặt tên Applet

Hầu hết các ứng dụng quen thuộc được đặt tên và xác định bởi một tên chuỗi. Tuy nhiên, trong công nghệ JavaCard, mỗi applet được xác định và lựa chọn sử dụng một "định danh ứng dụng" (AID). Ngoài ra, mỗi gói Java được gán một AID. Điều này có nghĩa là một gói khi nạp vào một thẻ được liên kết với các gói khác đã được đặt trên thẻ thông qua AID của chúng. Quy ước đặt tên này phù hợp với đặc tả thẻ thông minh được định nghĩa trong ISO 7816.

Một AID là một mảng các byte có thể được hiểu là hai mảng riêng biệt, như trong bảng 2.3 phần thứ nhất là giá trị 5 byte được gọi là RID (mã nguồn). Phần thứ hai là một giá trị có độ dài biến được gọi là PIX (thuộc tính mở rộng định danh). PIX có thể từ 0 đến 11 byte chiều dài. Do đó một AID có thể từ 5 đến 16 byte trong tổng chiều dài.

## Java Card Applet

JavaCard applet không nên nhầm lẫn với các applet Java chỉ vì chúng đều là các applet được đặt tên. Một JavaCard Applet là một chương trình Java tuân thủ một tập hợp các quy ước cho phép nó chạy trong môi trường chạy Java Card. Một JavaCard Applet không được dự định chạy trong môi trường trình duyệt. Lý do tên applet đã được chọn cho các ứng dụng Java Card được nạp vào môi trường chạy Java Card sau khi thẻ đã được sản xuất. Tức là, không giống như các ứng dụng trong nhiều hệ thống nhúng, các applet không cần phải ghi vào ROM trong quá trình sản xuất. Thay vào đó, họ có thể tự động tải xuống thẻ sau đó.

Lớp applet phải mở rộng từ lớp javacard.framework.Applet. Lớp Applet cơ bản là siêu lớp (super class) cho tất cả các applet cư trú trên một thẻ Java. Lớp applet là một kế hoạch chi tiết để giải thích các biến và các phương thức của một applet.

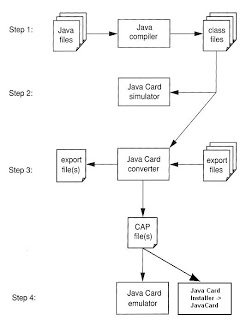
## Tiến trình phát triển Applet

Sự phát triển của một applet JavaCard bắt đầu cũng giống như với bất kỳ chương trình Java nào khác. Một nhà phát triển viết một hoặc nhiều lớp Java và biên dịch mã nguồn với một trình biên dịch Java, tạo ra một hoặc nhiều lớp. Hình 2.6 trình bày quá trình phát triển applet.

Tiếp theo, applet được chạy, thử nghiệm và gỡ lỗi trong một môi trường mô phỏng. Bộ mô phỏng mô phỏng môi trường chạy JavaCard trên máy PC hoặc một máy trạm khác.Trong môi trường mô phỏng, Applet chạy trên một máy ảo Java, và có các lớp của applet được thực thi. Theo cách này giả lập có thể sử dụng nhiều công cụ phát triển Java (máy ảo, trình gỡ lỗi, và các công cụ khác) và cho phép nhà phát triển kiểm tra hành vi của applet và nhanh chóng xem kết quả của applet không qua quá trình chuyển đổi. Trong bước này, các khía cạnh chức năng tổng thể của applet được kiểm tra. Tuy nhiên, một số tính năng thời gian chạy của máy ảo Java Card chẳng hạn như tường lửa applet và các hành của đối tượng, không thể được kiểm tra.

Sau đó, các tệp class của applet tạo nên một gói Java được chuyển đổi sang một tệp CAP bằng cách sử dụng JavaCard. Convertor – trình chuyển đổi thẻ Java không chỉ có đầu vào các tệp class được chuyển đổi mà còn là một hoặc nhiều tệp export. Khi gói applet được chuyển đổi, bộ chuyển đổi cũng có thể export ra một tệp export cho gói đó. Một tệp CAP fìle hoặc tệp export, export ra một gói Java. Nếu một applet bao gồm một số gói, một CAP fĩle và tệp export được tạo ra cho mỗi gói.

Trong bước tiếp theo, các tệp CAP đại diện cho applet được nạp và thử nghiệm trong môi trường mô phỏng. Trình mô phỏng cũng mô phỏng môi trường chạy JavaCard trên máy PC hoặc máy trạm. Tuy nhiên, trình mô phỏng là một công cụ thử nghiệm phức tạp hơn. Nó bao gồm một thực hiện máy ảo Java Card. Các hành vi của applet thực hiện trong giả lập nên được giống như hành vi của nó chạy trong một thẻ thực. Trong giai đoạn phát triển này, không chỉ applet được tiếp tục thử nghiệm. Mà còn là hành vi thời gian chạy của applet được đo. Trình gỡ lỗi cho phép nhà phát triển thiết lập các điểm ngắt hoặc chương trình đơn bước, theo dõi thời gian thực thi của sự thay đổi applet trong môi trường thời gian chạy của JavaCard đã mô phỏng hoặc giả lập. Khi applet được thử nghiệm và sẵn sàng để tải xuống một thẻ thực, applet, đại diện bởi một hoặc nhiều CAP, được tải và cài đặt trong thẻ thông minh Java.



Hình 2.6 Tiến trình phát triển Applet

## Cài đặt Applet

Việc cài đặt aplet được thực hiện tại nhà máy hoặc tại văn phòng của người phát hành và cũng có thực hiện sau khi phát hành, thông qua quá trình cài đặt an toàn (nếu nhà sản xuất thẻ xác định). Quá trình này bao gồm việc tải xuống một applet được ký kỹ thuật số, mà JCRE xác minh là hợp pháp, trước khi cài đặt applet. Các aplelet được cài đặt thông qua các tải không thể chứa các cuộc gọi phương thức tự nhiên do chúng không được tin cậy.

Applet gọi các phương thức tự nhiên phải được cài đặt tại nhà máy hoặc một môi trường tin cậy khác. Điều này được thực hiện vì lý do an ninh, vì các cuộc gọi nội bộ vượt qua khuôn khổ an ninh công nghệ Java và do đó phải được tin cậy cao trước khi được phép trên thẻ sau khi cài đặt JavaCard làm nền tảng. Không tương tác trực tiếp với thiết bị chấp nhận thẻ hoặc ứng dụng ngoại lệ. Các lớp được cài đặt có thể tương tác trực tiếp với JCRE hoặc với các lớp được cài đặt khác. JCRE chọn một applet và sau đó chuyển các APDU sang các applet đã chọn. Về bản chất, JCRE bảo vệ các nhà phát triển từ CPU thẻ thông minh, CAD và các giao thức truyền thông ISO cụ thể được sử dụng. JCRE cũng dịch các ngoại lệ không được ném ra bởi các lớp hoặc câu lệnh trả về bình thường trong các phương thức applet thành các giá trị trả về chuẩn ISO.

Kho lưu trữ cho một applet đã cài đặt không thể được phục hồi. Nếu một phiên bản mới hơn của applet được cài đặt, nó chiếm một vị trí lưu trữ mới và phiên bản trước đó của applet trở nên không thể truy cập được. Các applet thẻ Java cũng có thể được thực hiện không thể truy cập bằng cách loại bỏ tham chiếu của nó từ bảng đăng ký applet JCRE. Một khi các tài liệu tham khảo được gỡ bỏ, applet không còn có thể đạt được.

Việc cài đặt thẻ JavaCard làm cho các thành viên tĩnh của nó được khởi tạo. Công nghệ Java Card hỗ trợ khởi động tĩnh. Bộ khởi tạo không thể thực thi mã phần mềm Java, cũng không thể đặt thành viên tĩnh vào một giá trị không cố định (biến). Cài đặt cũng kết quả trong một cuộc gọi đến khởi tạo phương thức của applet (không giống như các applet Java).

Các ứng dụng đang chạy trong một thẻ thông minh Java giao tiếp với các ứng dụng máy chủ ở phía CAD bằng cách sử dụng APDU. Đối với mỗi lệnh APDU, applet đầu tiên giải mã giá trị của mỗi trường trong tiêu đề. Để thực thi các lệnh và đọc dữ liệu, applet có thể thực hiện các chức năng yêu cầu của lệnh bằng APDU. Đối với APDU phản ứng (command), applet cần định nghĩa một tập các từ trạng thái để cho biết kết quả xử lý lệnh APDU tương ứng. Trong quá trình xử lý thông thường, applet trả về từ trạng thái thành công. Nếu lỗi xảy ra, applet phải trả lại một từ trạng thái khác với thành công để biểu thị trạng thái bên trong của nó hoặc chẩn đoán lỗi.

Cài đặt Applet đề cập đến quá trình nạp các lớp applet trong một tệp CAP, kết hợp chúng với trạng thái thực thi của môi trường chạy Java Card, và tạo một ví dụ applet để đưa applet vào trạng thái có thể lựa chọn và thực thi. Trên nền tảng JavaCard, đơn vị tải và cài đặt là tệp CAP. Một tệp CAP bao gồm các lớp tạo nên gói Java. Một applet tối thiểu là một gói Java với một lớp duy nhất có nguồn gốc từ lớp javacard.framework Applet. Một applet phức tạp hơn với một số lớp có thể được tổ chức thành một gói Java hoặc một tập các gói Java.

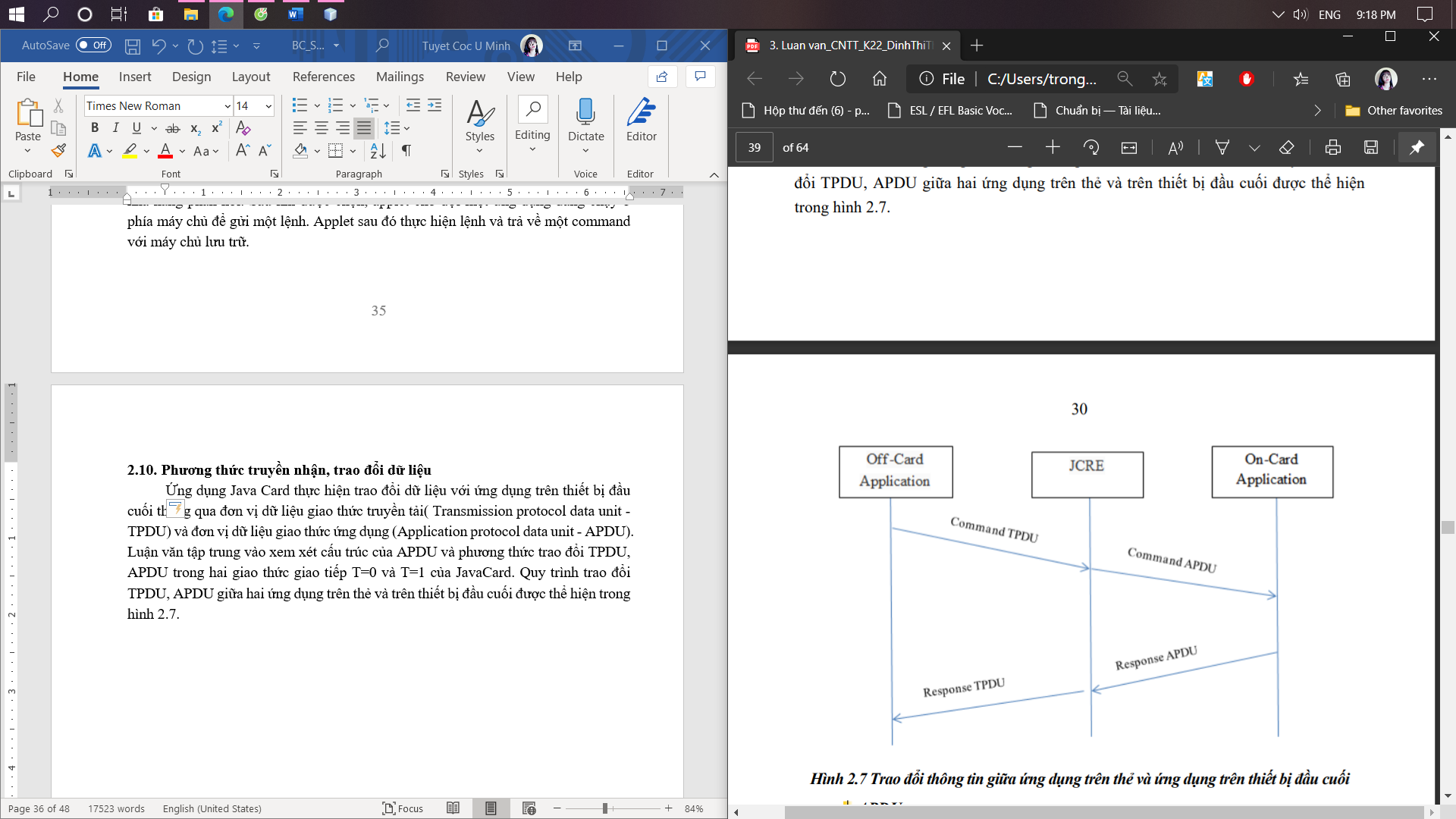
Để tải một applet, trình cài đặt thẻ sẽ thực hiện: thẻ sẽ lấy tệp CAP và chuyển đổi nó thành một chuỗi các lệnh APDU, mang nội dung tệp CAP. Bằng cách trao đổi các lệnh APDU với chương trình cài đặt không thẻ, trình cài đặt trên thẻ sẽ viết nội dung tệp CAP vào bộ nhớ liên tục của thẻ và liên kết các lớp trong tệp CAP với các class khác nằm trên thẻ. Trình cài đặt cũng tạo và khởi tạo bất kỳ dữ liệu nào được sử dụng nội bộ bởi JCRE để hỗ trợ applet này. Nếu applet yêu cầu một vài gói để chạy, mỗi tệp CAP được nạp vào thẻ.

Bước cuối cùng trong quá trình cài đặt applet, trình cài đặt tạo ra một ví dụ applet và đăng ký với JCRE. Để làm như vậy, trình cài đặt sẽ gọi phương thức cài đặt: public static void install (byte[] bArray, short offset, byte length)

Phương pháp cài đặt là một phương pháp của applet, tương tự như phương pháp chính trong một ứng dụng Java. Một applet phải thực hiện phương pháp cài đặt. Trong phương pháp cài đặt, nó gọi constructor của applet để tạo và khởi tạo một ví dụ applet. Các tham số cài đặt được gửi tới thẻ cùng với tệp CAP. Sau khi applet được khởi tạo và đăng ký với JCRE, nó có thể được chọn và chạy. JCRE xác định một applet đang chạy (một ví dụ applet), sử dụng AID. Applet này có thể tự đăng ký với JCRE bằng cách sử dụng mặc định AID được tìm thấy trong tệp CAP hoặc có thể chọn một tệp tin khác. Các thông số cài đặt có thể được sử dụng để cung cấp một AID thay thế JCRE là một môi trường đơn luồng. Điều này có nghĩa là chỉ một applet đang chạy cùng một lúc. Khi một applet được cài đặt lần đầu, nó ở trạng thái không hoạt động. Applet sẽ hoạt động khi nó được lựa chọn rõ ràng bởi một ứng dụng máy chủ lưu trữ Applet, giống như bất kỳ ứng dụng thẻ thông minh, là các ứng dụng có khả năng phản hồi. Sau khi được chọn, applet chờ đợi một ứng dụng đang chạy ở phía máy chủ để gửi một lệnh. Applet sau đó thực hiện lệnh và trả về một command với máy chủ lưu trữ.

## Phương thức truyền nhận, trao đổi dữ liệu

Ứng dụng Java Card thực hiện trao đổi dữ liệu với ứng dụng trên thiết bị đầu cuối thông qua đơn vị dữ liệu giao thức truyền tải( Transmission protocol data unit - TPDU) và đơn vị dữ liệu giao thức ứng dụng (Application protocol data unit - APDU). Luận văn tập trung vào xem xét cấu trúc của APDU và phương thức trao đổi TPDU, APDU trong hai giao thức giao tiếp T=0 và T=1 của JavaCard. Quy trình trao đổi TPDU, APDU giữa hai ứng dụng trên thẻ và trên thiết bị đầu cuối được thể hiện trong hình 2.7.

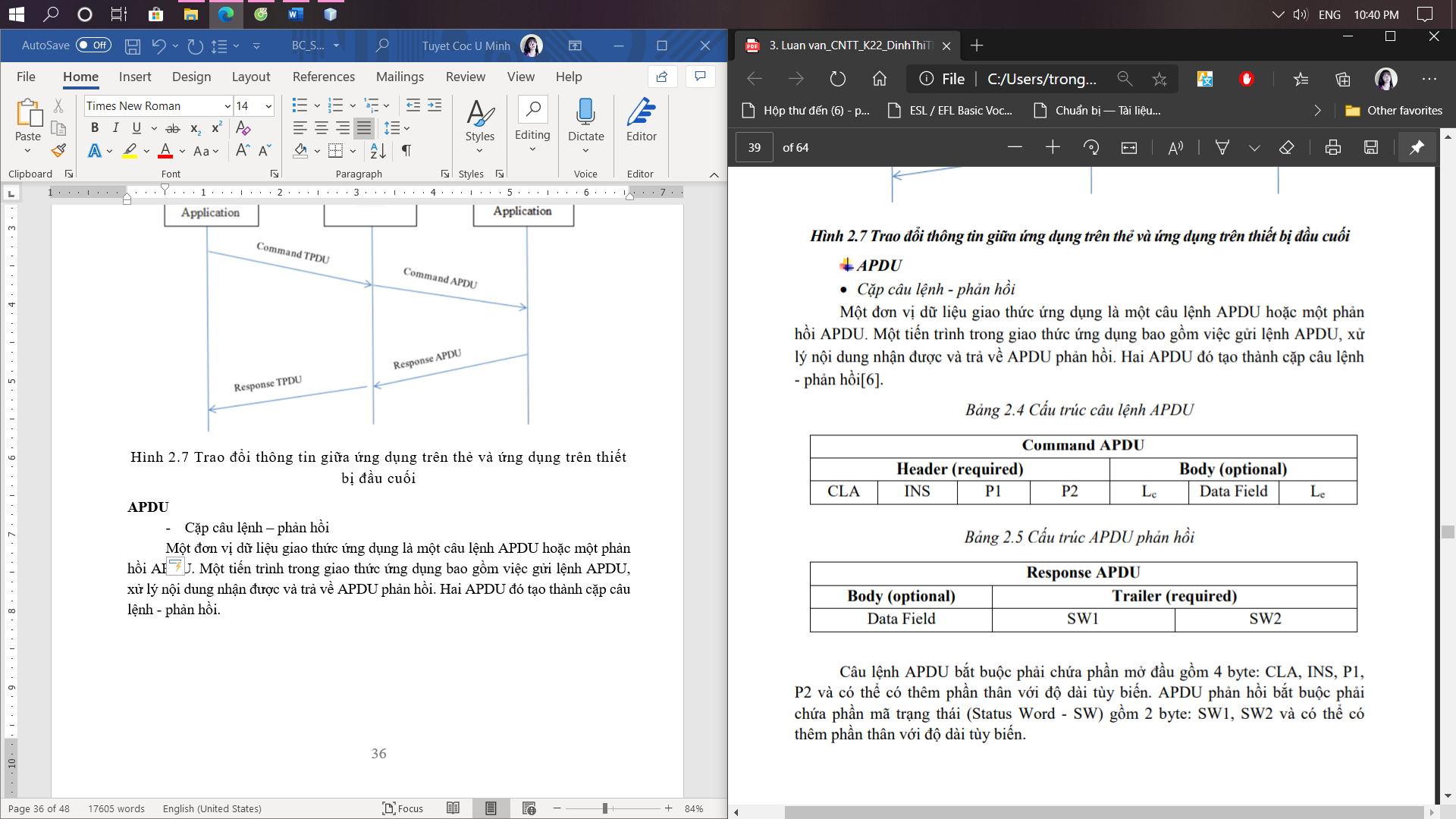


Hình 2.7 Trao đổi thông tin giữa ứng dụng trên thẻ và ứng dụng trên thiết bị đầu cuối

**APDU**

* Cặp câu lệnh – phản hồi

Một đơn vị dữ liệu giao thức ứng dụng là một câu lệnh APDU hoặc một phản hồi APDU. Một tiến trình trong giao thức ứng dụng bao gồm việc gửi lệnh APDU, xử lý nội dung nhận được và trả về APDU phản hồi. Hai APDU đó tạo thành cặp câu lệnh - phản hồi.



Câu lệnh APDU bắt buộc phải chứa phần mở đầu gồm 4 byte: CLA, INS, P1, P2 và có thể có thêm phần thân với độ dài tùy biến. APDU phản hồi bắt buộc phải chứa phần mã trạng thái (Status Word - SW) gồm 2 byte: SW1, SW2 và có thể có thêm phần thân với độ dài tùy biến.

Các trường dữ liệu trong cặp câu lệnh - phản hồi.

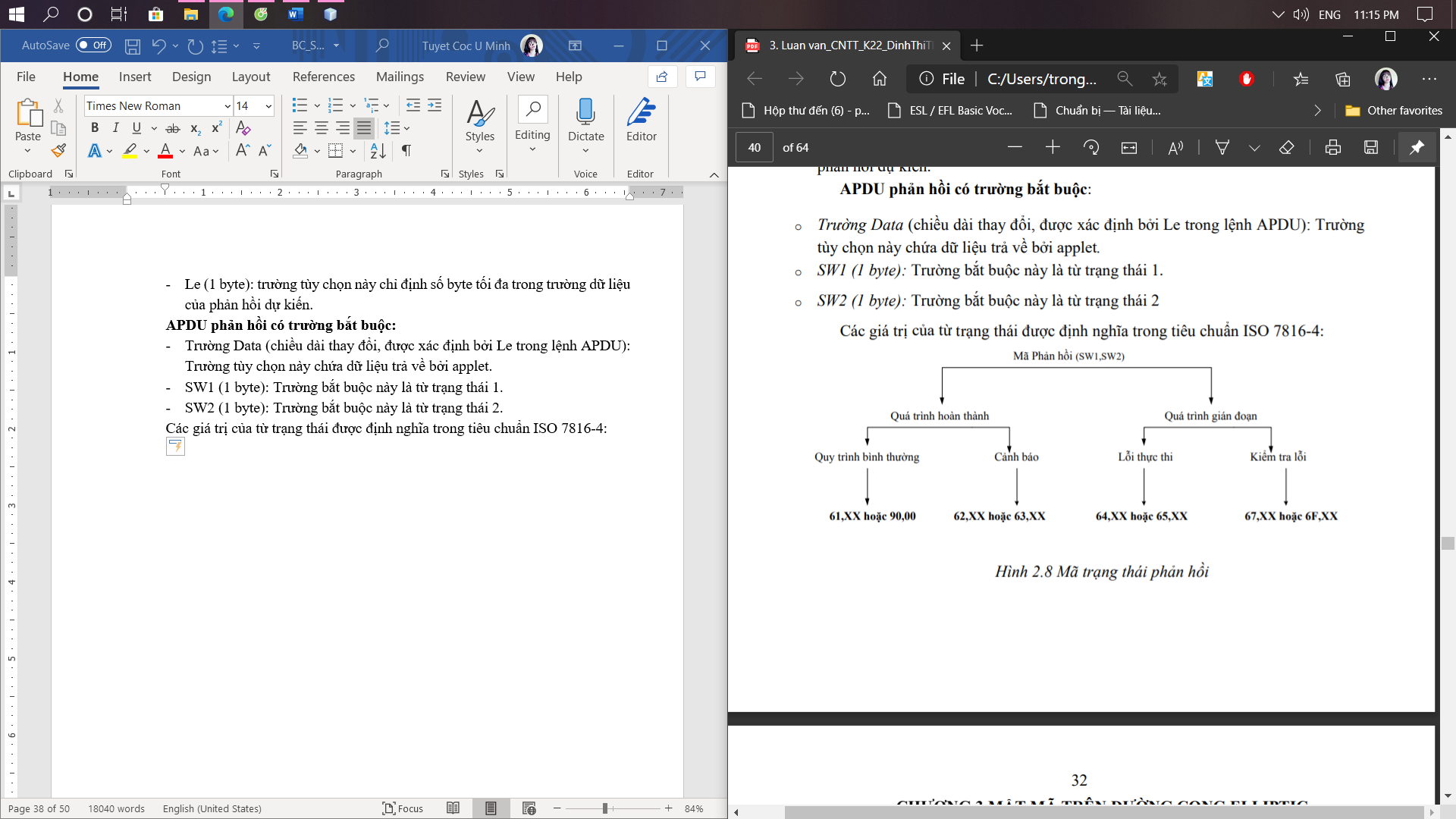
Mỗi một cặp câu lệnh - phản hồi có thể mang theo một trường dữ liệu câu lệnh/phản hồi.

* CLA(1 byte): Trường bắt buộc này xác định một lớp hướng dẫn cụ thể cho từng ứng dụng. Các giá trị CLA hợp lệ được định nghĩa trong tiêu chuẩn ISO 7816-4.
* INS(1 byte): Trường bắt buộc này chỉ ra một hướng dẫn cụ thể trong lớp hướng dẫn được xác định bởi trường CLA. Tiêu chuẩn ISO 7816-4 xác định các hướng dẫn cơ bản để sử dụng để truy cập vào dữ liệu trên thẻ khi nó được cấu trúc theo hệ thống tệp tin trên thẻ như được định nghĩa trong tiêu chuẩn. Các chức năng bổ sung đã được chỉ định ở nơi khác trong tiêu chuẩn, một số là các chức năng bảo mật. Bạn có thể xác định các giá trị INS riêng biệt cho ứng dụng của mình chỉ khi sử dụng một giá trị byte CLA thích hợp, theo tiêu chuẩn.
* P1(1 byte): trường bắt buộc này định nghĩa tham số chỉ dẫn 1. Bạn có thể sử dụng trường này để xác định trường INS, hoặc cho dữ liệu đầu vào.
* P2(1 byte): Trường bắt buộc này định nghĩa tham số chỉ dẫn 2. Bạn có thể sử dụng trường này để xác định trường INS, hoặc cho dữ liệu đầu vào.
* Lc (1 byte): trường tùy chọn này là số byte trong trường dữ liệu của lệnh.
* trường Data (biến, Lc số byte): trường tùy chọn này giữ dữ liệu lệnh.
* Le (1 byte): trường tùy chọn này chỉ định số byte tối đa trong trường dữ liệu của phản hồi dự kiến.

**APDU phản hồi có trường bắt buộc:**

* Trường Data (chiều dài thay đổi, được xác định bởi Le trong lệnh APDU): Trường tùy chọn này chứa dữ liệu trả về bởi applet.
* SW1 (1 byte): Trường bắt buộc này là từ trạng thái 1.
* SW2 (1 byte): Trường bắt buộc này là từ trạng thái 2.

Các giá trị của từ trạng thái được định nghĩa trong tiêu chuẩn ISO 7816-4:



Hình 2.8 Mã trạng thái phản hồi

# CHƯƠNG 3: HỆ MÃ RSA, VÀ THUẬT TOÁN MÃ HÓA AES

Chương này giới thiệu về hệ mật RSA, thuật toán mã hóa AES.

## Thuật toán RSA

Trong mật mã học, RSA là một thuật toán mật mã hóa khóa công khai. Đây là thuật toán đầu tiên phù hợp với việc tạo ra chữ ký điện tử đồng thời với việc mã hóa. Nó đánh dấu một sự tiến bộ vượt bậc của lĩnh vực mật mã học trong việc sử dụng khóa công cộng. RSA đang được sử dụng phổ biến trong thương mại điện tử và được cho là đảm bảo an toàn với điều kiện khóa đủ lớn. [RSA](http://people.csail.mit.edu/rivest/Rsapaper.pdf) là một hệ mã hóa bất đối xứng được phát triển bởi Ron Rivest, Adi Shamir và Leonard Adleman (tên của nó cũng chính là tên viết tắt của 3 tác giả này).

Thuật toán RSA có hai khóa: khóa công khai và khóa bí mật. Mỗi khóa là những số cố định sử dụng trong quá trình mã hóa và giải mã. Khóa công khai được công bố rộng rãi cho mọi người và được dùng để mã hóa. Những thông tin được mã hóa bằng khóa công khai chỉ có thể được giải mã bằng khóa bí mật tương ứng.

Hoạt động của RSA dựa trên 4 bước chính: Key generation (sinh khóa), Key distribution (chia sẻ key), Encription (mã hóa) và Decryption (giải mã).

**Bước 1: Key generation**

* Chọn ngẫu nhiên hai số nguyên tố p, q ( p khác q) được giữ bí mật
* Tính n = pq. n được dùng như một mô đun cho cả khóa công khai và khóa bí mật. Độ dài của nó thường được biểu diễn dạng bít là độ dài của khóa
* Tính giá trị hàm số ơ le Ø(n) = (p-1)(q-1)
* Chọn một số tự nhiên e sao cho 1< e < Ø(n) và là một số nguyên tố cùng nhau với Ø(n)
* Tính d sao cho d.e ≡ 1 (mod Ø(n)).

*Lưu ý:*

* Các số nguyên tố được chọn bằng phương pháp thử xác suất
* Các bước 4 và 5 có thể được thực hiện bằng giải thuật Euclid mở rộng
* Khóa công khia bao gồm:
  + Mô đun n
  + e là số mũ công khai (số mũ mã hóa)
* Khóa bí mật bao gồm:
  + Mô đun n
  + d là số mũ bí mật (số mũ giải mã)

**Bước 2: Key distribution**

Nếu A muốn gửi thông tin cho B và sử dụng mã hóa RSA thì A phải biết được public key để mã hóa tin nhắn của B và B phải dùng khóa bí mật để giải mã.

Để A có thể gửi tin nhắn đã mã hóa tới B. Thì B phải gửi khóa công khai (n,e) đến A thông qua một cách đáng tin cậy nhưng không cần thiết bí mật. Khóa bí mật của B là (d) không bao giờ được chia sẻ.

**Bước 3: Encryption**

Sau khi A nhận được khóa công khai của B thì A có thể gửi tin M tới B. Để làm được điều đó trước tiên A chuyển đổi M( tin nhắn văn bản, hình ảnh giọng nói…) thành một số nguyên m sao cho 0 ≤ m < n theo một hàm có thể đảo ngược ( tức là từ m có thể chuyển ngược lại M ) được thỏa thuận trước.

Lúc này A có m và biết n, e do B gửi tới. A sẽ tính c là bản mã hóa của m theo công thức: c = m­e mod n sau đó A gửi c đến B.

**Bước 4: Decryption**

B nhận được c từ A và biết khóa bí mật. B có thể tìm được m từ c theo công thức:

m= cd mod n

Quá trình giải mã hoạt động vì ta có:

cd≡ (m­e)d ≡ med (mod n)

Do e.d ≡ 1 nên med = m (mod p) và

med ≡ m (mod p)

và

med ≡ m (mod q)

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

med ≡ m (mod pq)

hay cd ≡ m (mod n)

biết m và B tìm lại M theo phương pháp mã hóa đã thỏa thuận trước.

Trong quá trình mã hóa và giải mã ta đều sử dụng bài toán: y= xz mod n. Để giải quyết bài toán ta phân tích z thành chuỗi số nhị phân như sau:

Zk-1zk-2zk-3….z2z1z0   
và ta có các bước tính toán sau:

x=y

for bit j=k-2 dowto 0

begin y=y\*y mod n

if zj ==1 then y=y\*x mod n end return

Thời gian tính toán của biểu thức này tỷ lệ thuận với số bit 1 trong số z. Trong quá trình mã hóa, để giảm khối lượng tính toán ta thường chọn e =3,17 hoặc 65537 (0x10001). Như ta thấy số bit 1 trong e chỉ là 2 số nên số lượng tính toán trong quá trình mã hóa rất ít. Tuy nhiên với thông số d, số bit 1 trong thông số này rất lớn nên làm cản trở cho việc tính toán theo phương pháp bình thường. Để giảm khối lượng tính toán, có một phương pháp thay thế hoàn hảo là sử dụng đặc trưng mới cho private key theo phương pháp CRT (chinese Remainder Theorem).

## Mã hóa RSA CRT

RSA CRT public key gồm 2 thông số, đó là mô đun n và số mũ e. Thông số mô đun n là tích của u số nguyên tố r, với u>=2, và thông số mô đun e là một số nguyên tố có giá trị nằm giữa 3 và n-1 thỏa mãn USCLN(e,y(n))=1 với y(n)=BSCNN (r0,r1,..,ru), với hai số nguyên tố đầu tiên r1 r2 lần lượt là q và p.

Private key không còn là (n,d) nữa mà bao gồm một quituple (p,q,dP,dQ,qlnv) và một (hoặc là không có) chuỗi các triple (ri,di,ti), với i = 3,…,u

Thông số mô đun n trong private key là tích của u số nguyên tố (u>=2). Thông số d là một số nguyên dương nhỏ hơn n thỏa mãn (e\*d) mod y(n) =1.

Các thành phần dP và dQ là những số nguyên dương nhỏ hơn p q thỏa mãn:

(e\*dP)mod(p-1) =1

(e\*dQ)mod(q-1)=1

Và thông số qlnv là số nguyên dương nhỏ hơn p thỏa mãn:

(q\*qlnv)mod p =1

Nếu u>2 thì chúng ta sẽ biểu diễn thêm các thành phần tripple (ri,di,ti) với i=3,…,u. Thông số ri là thành phần nhân trong tích tọa ra module n

Mỗi số mũ di thỏa mãn:

(e\*di)mod(ri-1)=1

Mỗi thông số ti là một số nguyên dương nhỏ hơn ri thỏa mãn:

(ri\*ti)mod ri =1 với ri=r1r2….ri-1

**Mã hóa:** Tính toán tương tự như quá trình mã hóa trong cơ chế bảo mật RSA thông thường.

**Giải mã:**

* Tính m1=cdP mod p và m2=cdQ mod q
* Nếu u>2, tính mi=cdi mod ri, i=3,…,u
* Tính h=(m1-m2)qlnv mod p
* Tính m=m2+qh
* Nếu u>2, đặt R=r1 và for i=3 to u do
  + R=Rri-1
  + Tính h=(mi-m) mod ri
  + Tính m=m2+Rh
* **Tính an toàn của RSA:** Tính an toàn của RSA chủ yếu dựa vào bộ tạo số ngẫu nhiên sinh ra 2 số nguyên tố p và q ban đầu. Việc tính ngược lại p và q từ n là chuyện hầu như không thể với hai số nguyên tố 2048 bits như đã đề cập ở trên. Nhưng việc tính ra d từ từ p và q là việc rất dễ dàng. Do đó nếu như một bên nào đó đoán ra được hoặc tìm ra lỗ hổng của bộ sinh số ngẫu nhiên đó thì coi RSA bị hoá giải.

## Thuật toán AES

1. Sơ lược về AES

Trong [mật mã học](https://vi.wikipedia.org/wiki/M%E1%BA%ADt_m%C3%A3_h%E1%BB%8Dc), Advanced Encryption Standard ([tiếng Anh](https://vi.wikipedia.org/wiki/Ti%E1%BA%BFng_Anh), viết tắt: AES, nghĩa là Tiêu chuẩn mã hóa tiên tiến) là một [thuật toán](https://vi.wikipedia.org/wiki/Thu%E1%BA%ADt_to%C3%A1n) [mã hóa khối](https://vi.wikipedia.org/wiki/M%C3%A3_h%C3%B3a_kh%E1%BB%91i) được [chính phủ](https://vi.wikipedia.org/wiki/Ch%C3%ADnh_ph%E1%BB%A7) [Hoa kỳ](https://vi.wikipedia.org/wiki/Hoa_k%E1%BB%B3) áp dụng làm tiêu chuẩn mã hóa.

AES được kỳ vọng áp dụng trên phạm vi thế giới và đã được nghiên cứu rất kỹ lưỡng. AES được chấp thuận làm tiêu chuẩn liên bang bởi [Viện tiêu chuẩn và công nghệ quốc gia Hoa kỳ](https://vi.wikipedia.org/w/index.php?title=Vi%E1%BB%87n_ti%C3%AAu_chu%E1%BA%A9n_v%C3%A0_c%C3%B4ng_ngh%E1%BB%87_qu%E1%BB%91c_gia_Hoa_k%E1%BB%B3&action=edit&redlink=1) (NIST) sau một quá trình tiêu chuẩn hóa kéo dài.

Thuật toán được thiết kế bởi hai nhà mật mã học người [Bỉ](https://vi.wikipedia.org/wiki/B%E1%BB%89): [Joan Daemen](https://vi.wikipedia.org/w/index.php?title=Joan_Daemen&action=edit&redlink=1) và [Vincent Rijmen](https://vi.wikipedia.org/w/index.php?title=Vincent_Rijmen&action=edit&redlink=1).

1. Mô tả thuật toán:

AES chỉ làm việc với các khối dữ liệu (đầu vào và đầu ra) 128 bít và [khóa](https://vi.wikipedia.org/wiki/Kh%C3%B3a_(m%E1%BA%ADt_m%C3%A3)) có độ dài 128, 192 hoặc 256 bít

Mỗi khối dữ liệu 128 bit đầu vào được chia thành 16 byte, có thể xếp thành 4 cột, mỗi cột 4 phần tử hay là một [ma trận](https://vi.wikipedia.org/wiki/Ma_tr%E1%BA%ADn) 4x4 của các byte, nó được gọi là ma trận trạng thái. Trong quá trình thực hiện thuật toán các toán tử tác động để biến đổi ma trận trạng thái này.

1. Các bước thực hiện thuật toán

Bao gồm các bước:

1. *Khởi động vòng lặp*
   * *AddRoundKey* — Mỗi cột của trạng thái đầu tiên lần lượt được kết hợp với một khóa con theo thứ tự từ đầu dãy khóa.
2. *Vòng lặp*
   * *SubBytes* — đây là phép thế (phi tuyến) trong đó mỗi byte trong trạng thái sẽ được thế bằng một byte khác theo bảng tra ([Rijndael S-box](https://vi.wikipedia.org/w/index.php?title=Rijndael_S-box&action=edit&redlink=1)).
   * *ShiftRows* — dịch chuyển, các hàng trong trạng thái được dịch vòng theo số bước khác nhau.
   * *MixColumns* — quá trình trộn làm việc theo các cột trong khối theo một phép biến đổi tuyến tính.
   * *AddRoundKey*
3. *Vòng lặp cuối*
   * *SubBytes*
   * *ShiftRows*
   * *AddRoundKey*

Tại chu trình cuối thì bước MixColumns không thực hiện.

*Bước SubBytes*

Các byte được thế thông qua bảng tra [S-box](https://vi.wikipedia.org/w/index.php?title=Rijndael_S-box&action=edit&redlink=1). Đây chính là quá trình phi tuyến của thuật toán. Hộp S-box này được tạo ra từ một [phép biến đổi khả nghịch](https://vi.wikipedia.org/w/index.php?title=Ph%C3%A9p_bi%E1%BA%BFn_%C4%91%E1%BB%95i_kh%E1%BA%A3_ngh%E1%BB%8Bch&action=edit&redlink=1) trong [trường hữu hạn](https://vi.wikipedia.org/w/index.php?title=Tr%C6%B0%E1%BB%9Dng_h%E1%BB%AFu_h%E1%BA%A1n&action=edit&redlink=1) 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 bước nhất định. Đối với AES, 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 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 bốn byte đầu ra. Cùng với bước ShiftRows, MixColumns đã tạo ra tính chất [khuếch tán](https://vi.wikipedia.org/w/index.php?title=Khu%E1%BA%BFch_t%C3%A1n_(m%E1%BA%ADt_m%C3%A3)&action=edit&redlink=1) cho thuật toán. Mỗi cột được xem như một đa thức trong trường hữu hạn và được nhân với đa thức {\displaystyle c(x)=3x^{3}+x^{2}+x+2}(modulo {\displaystyle x^{4}+1}). Vì thế, bước này có thể được xem là phép [nhân](https://vi.wikipedia.org/w/index.php?title=Nh%C3%A2n_(to%C3%A1n_h%E1%BB%8Dc)&action=edit&redlink=1) [ma trận](https://vi.wikipedia.org/wiki/Ma_tr%E1%BA%ADn) trong trường hữu hạn.

*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](https://vi.wikipedia.org/w/index.php?title=XOR&action=edit&redlink=1) từng bít của khóa con với khối dữ liệu.

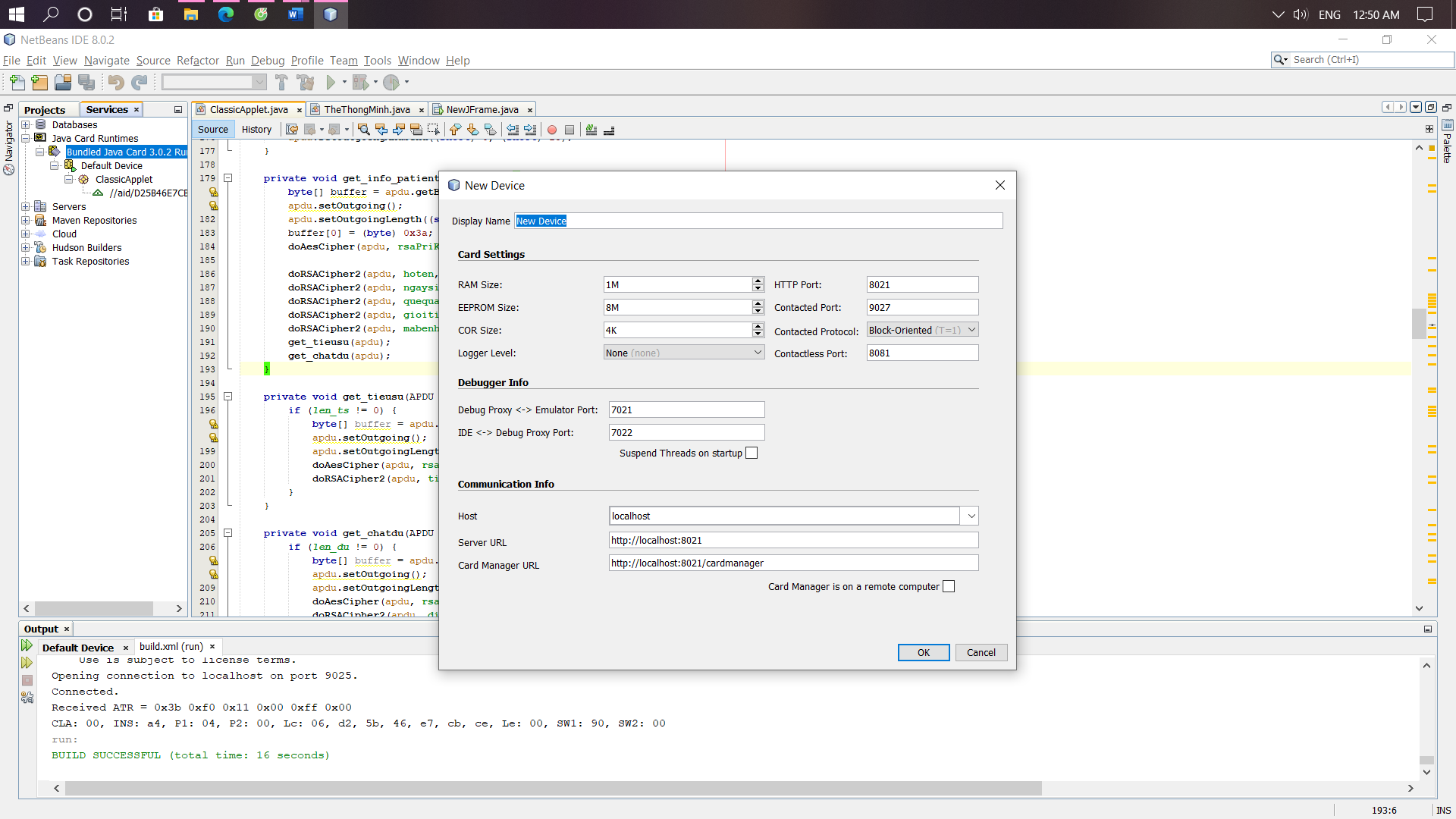
# CHƯƠNG 4: ỨNG DỤNG CÔNG NGHỆ JAVACARD XÂY DỰNG ỨNG DỤNG THẺ KHÁM BỆNH TRÊN NETBEAN IDE

Chương này trình bày việc triển khai ứng dụng thành 2 thành phần, thành phần ứng dụng applet dùng để cài đặt lên smart card và thành phần thứ 2 là ứng dụng người dùng với giao diện để người dùng giao tiếp với applet. Ngoài ra trong phạm vi nghiên cứu phát triển, ứng dụng người dùng có hiển thị các apdu lệnh và apdu phản hồi từ applet.

## Giao tiếp giữa javacard applet và netbean

## Java Card Runtimes

Để có thể giao tiếp với javacard applet, netbean cung cấp cho chúng ta một service là Java Card Runtimes. Java Card Runtimes cung cấp cho chúng ta một thiết bị ảo để cài đặt applet và thực hiện việc kết nối từ ứng dụng người dùng tới applet thông qua socket. Hình 4.1 cho ta thấy cấu hình của thiết bị card ảo của Java Card Runtimes.

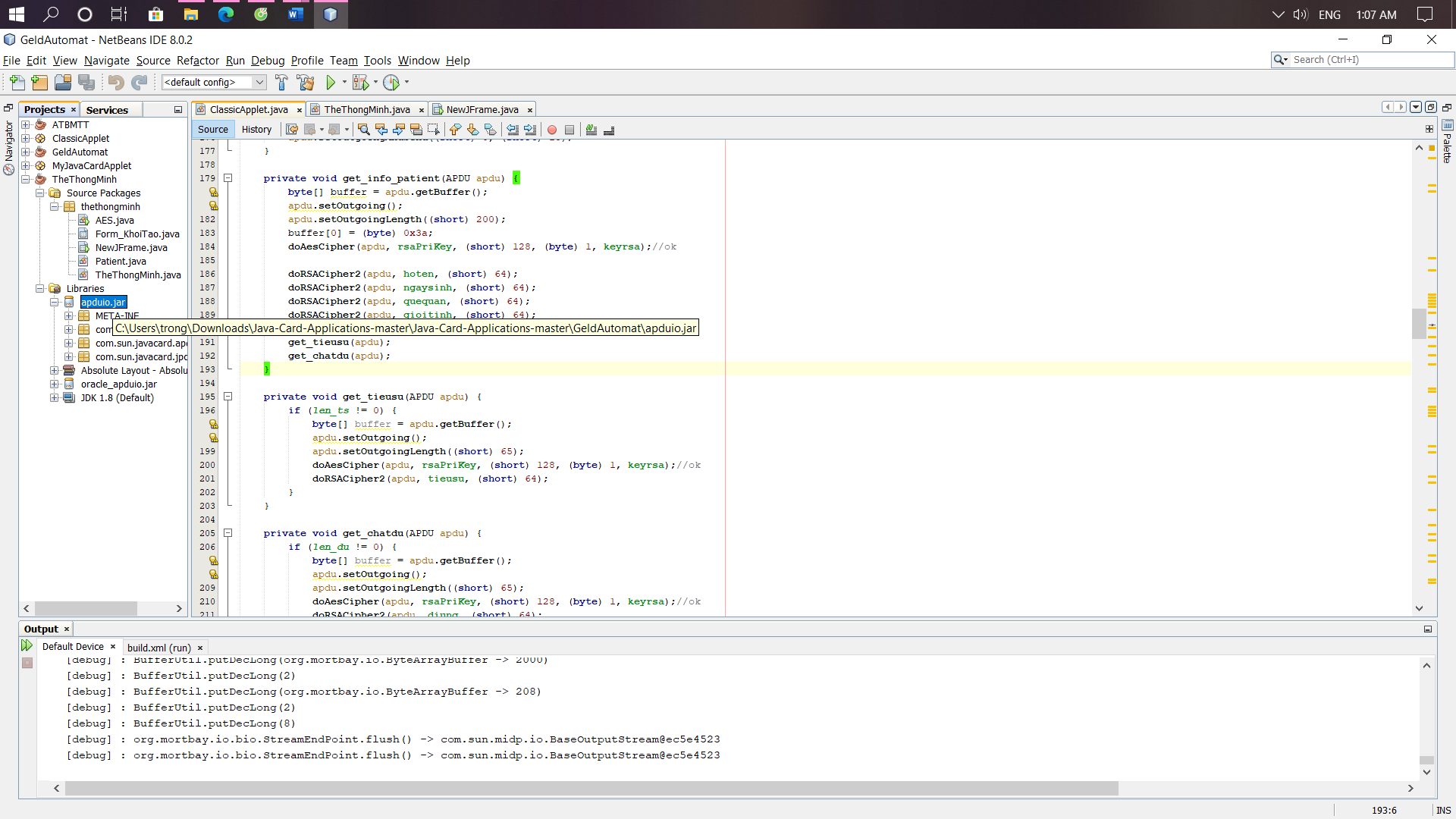


Hình 4.1. Cấu hình của thiết bị card ảo.

Tham số đáng chú ý là Contacted Port, đây là port của socket mà chúng ta sẽ thực hiện việc giao tiếp giữa chương trình người dùng và applet.

## Apdu in out library

Để có thể gửi các lệnh Apdu tới applet, ứng dụng người dùng trên netbean cần tạo ra đối tượng APDU thông qua thư viện apduio.jar được cung cấp. Chúng ta cần thêm thư viện này vào project của chúng ta trước khi triển khai viết các chức năng cho ứng dụng.



Là một thư viện đi kèm với bộ công cụ phát triển Java Card (Java card development kit). APDU I/O được sử dụng để phát triển các ứng dụng thẻ Java và giả lập thiết bị đọc thẻ . Cung cấp các phương thức để trao đổi lệnh APDU bằng cách sử dụng giao thức T = 0 qua TLP224 và T=1 qua sử dụng API PC/SC.

Thư viện APDU I/O nằm trong tệp lưu trữ Java (JAR) apduio.jar

Hoạt động giao tiếp giữa thẻ và thiết bị giả lập thông qua socket với cổng được đặt mặc định hoặc thiết lập lại.

## Thiết kế phần mềm

## Biểu đồ phân rã chức năng



Hình 4.2. Biểu đồ phân rã chức năng

*Chức năng quản lý thẻ*

* + Ban đầu để thực hiện các yêu cầu trên thẻ thì người dùng phải nhập đúng mã PIN nhưng có hạn chế số lần nhập mã PIN nếu nhập sai quá ba lần thì thẻ sẽ bị khoá (Block)
  + Người dùng sử dụng chức năng Unblock để mở khoá thẻ

*Chức năng quản lý dữ liệu trên thẻ*

* Khởi tạo dữ liệu thẻ: Với thẻ mới hoặc thẻ muốn khởi tạo lại thông tin, chương trình cung cấp chứ năng khởi tạo dữ liệu để thêm mới dữ liệu
* Cập nhật dữ liệu thẻ: Sức khoẻ bệnh nhân có thể thay đổi theo từng giai đoạn với mỗi lần khám bệnh thì bệnh nhân sẽ được cập nhật lại thông tin do đó. chương trình cung cấp hai trường mà thẻ có thể cập nhật đó là tiểu sử bệnh nhân và chất gây dị ứng
* Xoá dữ liệu: Để khởi tạo lại thông tin mà thẻ đã có sẵn thì trước đó người dùng phải xoá dữ liệu đã có
* Gửi dữ liệu lên Applet: Sau khi khởi tạo thông tin bệnh nhân dữ liệu sẽ được gửi đến Applet tại đây dữ liệu được xử lý và hiển thị lên giao diện thẻ

## Biểu đồ hoạt động

* + 1. *Khi thẻ không chứa dữ liệu hoặc khởi tạo lại dữ liệu*



Hình 4.3. Biểu đồ luồng hoạt động thẻ khi chưa có dữ liệu

Đối với một thẻ mới chưa có dữ liệu người dụng nạp thẻ vào thiết bị chấp nhận thẻ gọi đến hàm thiết lập kết nối thông qua phương thức socket(). Sau đó nạp dữ liệu lên thẻ và gửi dữ liệu đến applet thực hiện cách trao đổi với applet, cuối cùng sau khi thực hiện truy vấn xong gọi hàm ngắt kết nối.

* + 1. *Biểu đồ luồng hoạt động khi thẻ đã có dữ liệu*

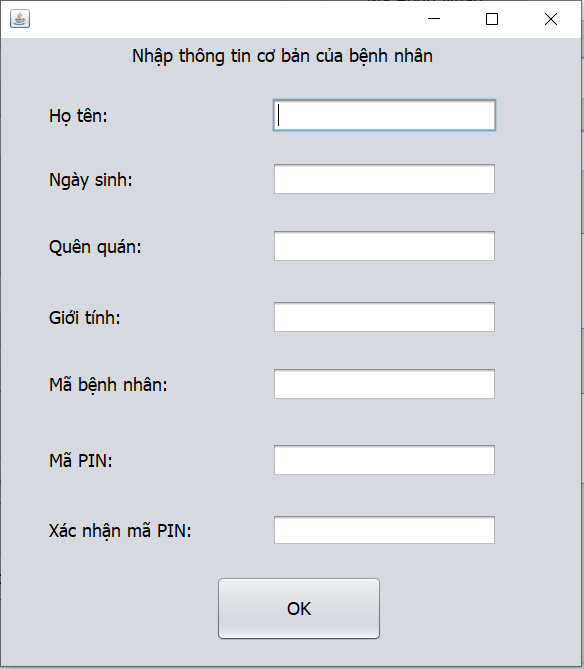


Hình 3. Biểu đồ luồng hoạt động của thẻ đã có dữ liệu

Bắt đầu ta thiết lập kết nối thông qua socket, tiến hành nhập mã PIN nếu sai quá ba lần sẽ bị khoá thẻ. Sau khi đăng nhập thành công thông tin bệnh nhân sẽ được hiển thị lên giao diện Applet. Người dùng thao tác với thẻ và gọi hàm ngắt kết nối để kết thúc

## 4.3. Thiết kết giao diện

1. *Giao diện nhập thông tin*



Hình 4.4. Giao diện nhập thông tin bệnh nhân

Khi thẻ chưa có dữ liệu hoặc khởi tạo lại thông tin thẻ ta sử dụng giao diện nhập thông tin. Bao gồm các thông tin cơ bản như hình 4.

1. *Giao diện lệnh APDU*

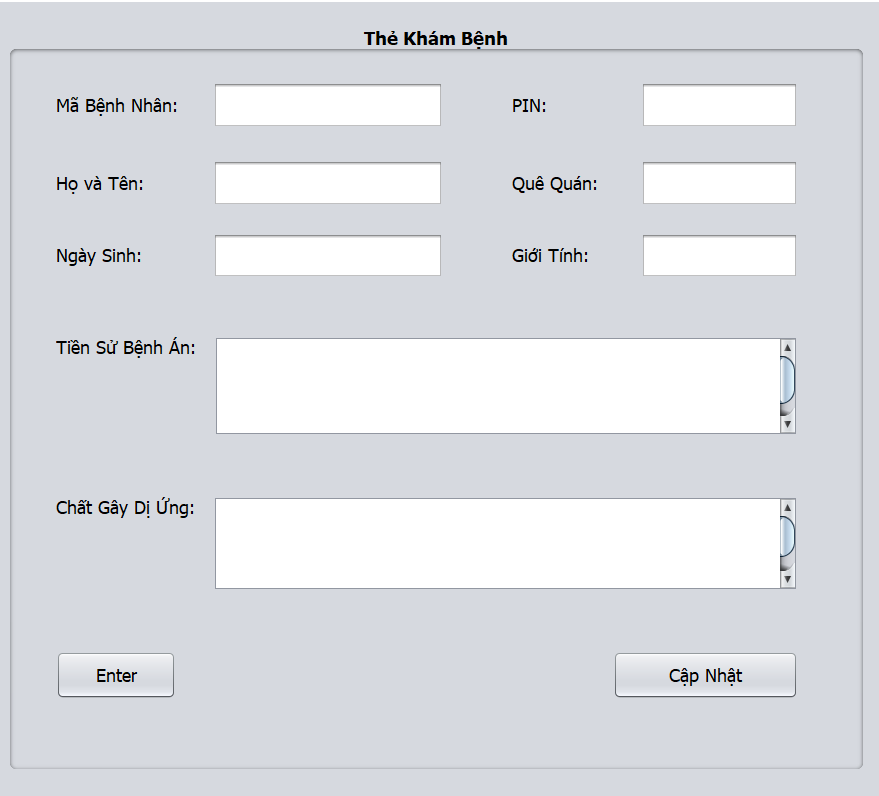


Hình 4.5. Giao diện APDU lệnh

Để mô tả quá trình gửi nhận lệnh từ thiết bị lên thẻ. Giao diện cho thấy các lệnh APDU mỗi lần thực hiện một yêu cầu đồng thời có các chức năng khác như mở khoá thẻ, khởi tạo lại thẻ, gửi dữ liệu lên Applet và xoá dữ liệu thẻ.

Ngoài ra trên giao diện APDU còn có chức năng thiết lập kết nối và huỷ kết nối đến thẻ.

1. *Giao diện hiển thị thông tin bệnh nhân*



Hình 4.6. Giao diện hiển thị thông tin bệnh nhân

Ngoài hiển thị thông tin bệnh nhân còn có thêm hai trường tiền sử bệnh án và chất gây dị ứng. Đây là hai trường thông tin có thể cập nhật lại sau mỗi lần khám bệnh của bệnh nhân

**4.4 Phát triển Applet**

**4.4.1.** **Hoạt động mã hoá**

*- Bước 1*: Tạo cặp khóa trong mã hóa RSA

Sau khi nhận thông tin bệnh nhân đưa xuống thẻ thực hiện gọi hàm sinh khoá tự động sử dụng thuật toán RSA. Đầu tiên tạo ra cặp khoá RSA trước bằng cách sinh ngẫu nhiên với số mũ e là (10001), thành phần mô đun n trong khóa RSA được sử dụng có độ dài 512bit. Sau đó lưu cặp khóa được sinh ra này lại vào 2 đối tượng private và public key.

* *Bước 2:* Dùng khoá công khai của thuật toán RSA để mã hoá thông tin bệnh nhân.
* *Bước 3:* Mã hóa khóa bí mật của mật mã RSA bằng thuật toán AES

Sử dụng mã PIN của người dùng làm khoá thuật toán AES và thực hiện mã hoá khoá bí mật của thuật toán RSA. Và lưu trữ private key của thuật toán RSA sau khi được sinh ngẫu nhiên ở bên trên thành chuỗi sau khi được mã hóa AES.

* *Bước 4:* Lưu trữ các khoá thuật toán, thông tin bệnh nhân đã được mã hoá trên thẻ.
  + 1. **Giải mã dữ liệu**
* *Bước 1*: Người dùng nhập mã PIN

Người dùng nhập mã PIN vào giao diện, sau đó APDU sẽ gửi mã PIN tới applet, đầu tiên applet thực hiện kiểm tra xem trạng thái của thẻ và mã PIN đó có khớp với thông tin PIN được lưu trong thẻ hay không.

* *Bước 2*: Giải mã private key của mã hóa RSA

Applet sử dụng mã PIN để làm đầu vào của khóa AES, sau đó giải mã khóa private key đã được mã hóa ra một đối tượng private key khác.

* *Bước 3:* Dùng khoá private được giải mã này để giải mã thông tin người dùng
* *Bước 4:* Gửi thông tin giải mã được lên giao diện thẻ.

Sau khi hoàn tất giải mã (thông qua gọi hàm doFinal) thì sẽ phản hồi chuỗi byte giải mã ra được ra APDU. APDU phân tích các byte được applet gửi ra rồi hiển thị lên màn hình ứng dụng.

# KẾT LUẬN

**Các kết quả đạt được:**

* Giới thiệu tổng quan về thẻ thông minh: khái quát lịch sử phát triển của thẻ thông minh, nêu lên cấu tạo và phân loại thẻ. Phân tích chi tiết về ưu nhược điểm của thẻ thông minh, ngoài ra thách thức trong việc phát triển thẻ thông minh.
* Nghiên cứu về công nghệ Java Card:
  + Giới thiệu về JavaCard, kiến trúc của nó, tập ngôn ngữ.
  + Trình bày về máy ảo để chạy, môi trường chạy, api java card, quy ước đặt tên, ứng dụng applet
  + Trình bày về các giao thức truyền nhận dữ liệu giữa thẻ và thiết bị đầu cuối
* Nghiên cứu về hệ mã RSA và thuật toán mã mã hóa AES.
* Nghiên cứu phát triển úng dụng applet trên Netbean.
* Đã phát triển được một ứng dụng applet mã hóa và giải mã thông tin sử dụng hệ mã RSA.

**Hướng phát triển**

* Nâng cao độ dài khóa của thuật toán RSA.
* Tiếp tục phát triển thêm để hoàn thiện tính năng của ứng dụng.