# Chương V QUẢN LÝ KHÓA

## 5.1 Khóa phiên là gì?

Một khoá phiên là một đơn sử dụng chìa khóa đối xứng được sử dụng để mã hóa tất cả các thông điệp trong một phiên truyền thông . Một thuật ngữ có liên quan chặt chẽ là khóa mã hóa nội dung ( CEK ), khóa mã hóa lưu lượng ( TEK ) hoặc khóa đa hướng đề cập đến bất kỳ khóa nào được sử dụng để mã hóa tin nhắn, trái với các mục đích sử dụng khác như mã hóa các khóa khác ( khóa mã hóa khóa ( KEK ) hoặc gói khóa phím ).

## 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 như thế nào?

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*

Cách khắc phục:

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*

## 5.4. Giao thức Kerberoslà gì?

Kerberos là một giao thức mật mã dùng để xác thực trong các mạng máy tính hoạt động trên những đường truyền không an toàn. Giao thức Kerberos có khả năng chống lại việc nghe lén hay gửi lại các gói tin cũ và đảm bảo tính toàn vẹn của dữ liệu. Mục tiêu khi thiết kế giao thức này là nhằm vào mô hình máy chủ-máy khách (client-server) và đảm bảo nhận thực cho cả hai chiều.

Giao thức được xây dựng dựa trên mật mã hóa khóa đối xứng và cần đến một bên thứ ba mà cả hai phía tham gia giao dịch tin tưởng.

## 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.

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

## 5.6 Cách thiết lập một khoá phiên đối xứng cơ bản?

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.

## 5.7 Cách thiết lập một khoá phiên đối xứng băng 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*)) .

## 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 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).

## 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.

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.

## 6.3 Các loại tấn côg

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 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

## 6.4 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).

*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

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.

## 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

* 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.