Đề cương ôn tập

Contents

[1 Câu hỏi lý thuyết 2](#_Toc89675260)

[1.1 Trình bày các hình thức tấn công? Nêu ví dụ minh họa? Liên hệ với các mục tiêu của an ninh mạng 2](#_Toc89675261)

[1.2 Trình bày các mục tiêu của an ninh mạng? Lấy ví dụ? 2](#_Toc89675262)

[1.3 Khái niệm mã hóa công khai? Sử dụng mã hóa công khai làm chữ ký số? 3](#_Toc89675263)

[1.4 Trình bày khái niệm firewall, cơ chế kiểm soát, ưu điểm và hạn chế? 3](#_Toc89675264)

[1.5 Khái niệm về các phần mềm độc hại, phân loại? 4](#_Toc89675265)

[1.6 Phân loại firewall? Trình bày về cơ chế Package Filtering? 5](#_Toc89675266)

[1.7 Nêu các hình thức tấn công? Liên hệ với các mục tiêu của an ninh mạng(Giống câu 1) 5](#_Toc89675267)

[1.8 Trình bày về Circuit‐level Gateway trong firewall ? 5](#_Toc89675268)

[1.9 Trình bày về các loại firewall topology? 6](#_Toc89675269)

[1.10 Khái niệm chữ ký số? Sử dụng mã hóa công khai thực hiện giao dịch có chữ ký số? 6](#_Toc89675270)

[1.11 Trình bày các bước để mã hóa theo thuật toán Caesar Cipher? Lấy ví dụ mã hóa một chuỗi ký tự tùy chọn để minh họa. 7](#_Toc89675271)

[1.12 Khi trao đổi thông điệp qua mạng tính toàn vẹn (integrity) được yêu cầu thế nào? Thực hiện yêu cầu toàn vẹn bằng giải pháp nào? 7](#_Toc89675272)

[1.13 Trình bày các bước để mã hóa theo thuật toán Vigenere Cipher? Lấy ví dụ mã hóa một chuỗi ký tự tùy chọn để minh họa. 8](#_Toc89675273)

[1.14 Nêu những hạn chế (nguy cơ bị bẻ khóa) của các phương pháp mã hóa kinh điển. Lấy ví dụ trong trường hợp Caesar Cipher và Vigenere Cipher 9](#_Toc89675274)

[1.15 Trên cơ sở phân tích các hạn chế của các thuật toán mã hóa cổ điển (Caesar Cipher, Vigeniar Cipher), các chuẩn mã hóa tiên tiến được thiết kế như thế nào? Trình bày các phép biến đổi cơ bản hay được sử dụng trong các chuẩn mã hóa tiên tiến. 9](#_Toc89675275)

[1.16 Nêu các hạn chế của firewall? Các cơ chế thiết lập firewall? 11](#_Toc89675276)

[1.17 Trình bày cách thiết lập một bộ khóa trong thuật toán RSA. Sử dụng bộ khóa đó để mã hóa và giải mã một ký tự tùy chọn 12](#_Toc89675277)

[1.18 Nêu những hiểu biết của bạn về hàm băm (Hash function) và sử dụng hàm băm trong đảm bảo tính toàn vẹn của thông điệp? 13](#_Toc89675278)

[1.19 Nêu kiến trúc các mô hình cài đặt firewall? 13](#_Toc89675279)

[1.20 Trình bày về SQL Injection và cách khắc phục 14](#_Toc89675280)

[2 Bài tập 15](#_Toc89675281)

[2.1 Sử dụng Java Cryptography Architecture (JCA), viết class SymmetricCryp để thực hiện các chức năng mã hóa và giải mã đối xứng theo thuật toán AES gồm các hàm 15](#_Toc89675282)

[2.1.1 Sinh khóa đối xứng 15](#_Toc89675283)

[2.1.2 Tạo khóa đối xứng từ một chuỗi cho trước 15](#_Toc89675284)

[2.1.3 Mã hóa thông điệp có 2 tham số: Tham số 1 là chuỗi cần mã hóa, tham số thứ 2 là khóa được tạo ở 1.1 16](#_Toc89675285)

[2.1.4 Mã hóa thông điệp có 2 tham số: Tham số 1 là chuỗi cần mã hóa tham số 2 là khóa được tạo ở 1.2 16](#_Toc89675286)

[2.1.5 Giải mã có 2 tham số: Tham số 1 là bản mã, tham số thứ 2 là khóa bí mật được tạo ở 1.1 16](#_Toc89675287)

[2.1.6 Giải mã có 2 tham số: Tham số 1 là bản mã, tham số thứ 2 là khóa bí mật được tạo ở 1.2 16](#_Toc89675288)

[2.1.7 Sử dụng class SymetricCryp để mã hóa một chuỗi cho trước hoặc giải mã ra bản rõ từ bản mã cho trước. 17](#_Toc89675289)

[2.2 Sử dụng Java Cryptography Architecture (JCA) để viết các hàm băm thông điệp cho trước theo thuật toán: 17](#_Toc89675290)

[2.2.1 MD5 17](#_Toc89675291)

[2.2.2 SHA1 17](#_Toc89675292)

[2.2.3 SHA-256 18](#_Toc89675293)

[2.3 Sử dụng Java Crypto Architecture (JCA), viết chương trình mã hóa và giải mã sử dụng mã hóa công khai RSA: Viết class RSAKeyGen để sinh bộ khóa công khai gồm các hàm 19](#_Toc89675294)

[2.3.1 Sinh bộ khóa theo thuật toán RSA 19](#_Toc89675295)

[2.3.2 Lưu bộ khóa ra file 19](#_Toc89675296)

[2.3.3 Hàm đọc file để nạp khóa bí mật, tham số truyền vào là tên file 20](#_Toc89675297)

[2.3.4 Hàm đọc file để nạp khóa công khai, tham số truyền vào là tên file 20](#_Toc89675298)

[2.3.5 Hàm mã hóa một thông điệp cho trước bằng khóa bí mật, tham số truyền vào gồm tham số thứ nhất là thông điệp cần mã hóa, tham số thứ 2 là khóa bí mật 20](#_Toc89675299)

[2.3.6 Hàm giải mã thông điệp bằng khóa công khai, tham số truyền vào gồm tham số thứ nhất là bản mã cần giải mã, tham số thứ 2 là khóa công khai. 21](#_Toc89675300)

[2.3.7 Sử dụng lớp RSACryp để mã hóa một chuỗi và băm chuỗi theo một thuật toán băm cho trước rồi lưu cả bản mã và bản băm vào một file 21](#_Toc89675301)

[2.3.8 Đọc file chứa bản mã và bản băm của một thông điệp, sử dụng lớp RSACryp để giải mã ra thông điệp gốc và băm lại bằng thuật toán băm cho trước. Sau đó đem so sánh hai bản băm để xác nhận tính toàn vẹn 22](#_Toc89675302)

[2.4 Lập trình mô tả lỗi SQL Injection và cách khắc phục 23](#_Toc89675303)

# Câu hỏi lý thuyết

## Trình bày các hình thức tấn công? Nêu ví dụ minh họa? Liên hệ với các mục tiêu của an ninh mạng

* Tấn công gián đoạn (ngăn chặn thông tin) – interruption: Tài nguyên thông tin bị phá hủy, không sẵn sàng phục vụ hoặc không sử dụng được. Đây là hình thức tấn công làm mất khả năng sẵn sàng phục vụ của thông tin
* VD: A gửi cho B 1 gói tin bị chặn bởi C và gói tin đấy ko đến được tới B.
* Liên hệ: Availability
* Tấn công chặn bắt thông tin – interception: Kẻ tấn công có thể truy nhập tới tài nguyên thông tin. Dữ liệu bị truy xuất bởi người không có thẩm quyền. Đây là hình thức tấn công vào tính bí mật của thông tin
* VD: A gửi cho B gói tin bị C xem nhưng gói tin vẫn đến được tới B
* Liên hệ: confidentiality
* Tấn công sửa đổi thông tin – modification: Kẻ tấn công chèn các thông tin và dữ liệu giả vào hệ thống. Làm thay đổi bản tin truyền thông giữa A và B thay đổi bởi người không có thẩm quyền -> mất tính toàn vẹn bản tin trao đổi giữa 2 người. Đây là hình thức tấn công vào tính toàn vẹn của thông tin
* VD: A gửi cho B 1 gói tin, C chặn gói tin sửa đổi nội dung rồi gửi gói tin đã bị chỉnh sửa đến B
* Liên hệ: integrity
* Tấn công chèn thông tin giả mạo – fabrication: Kẻ tấn công chèn các thông tin và dữ liệu giả vào hệ thống. thông tin sai sự thật đc tạo ra và gửi đến địa chỉ chủ đích, phá vỡ yêu cầu nguồn gốc của thông điệp. Đây là hình thức tấn công vào tính xác thực của thông tin
* VD: C giả mạo A gửi gói tin đến B
* Liên hệ: message origin authentication, Timeliness

## Trình bày các mục tiêu của an ninh mạng? Lấy ví dụ?

Có 3 mục tiêu chủ yếu:

* Tính bảo mật. Ví dụ: A và B ko muốn thông điệp của họ bị đọc bởi người khác.
* Tính toàn vẹn. Ví dụ: A và B ko muốn thông điệp của họ bị thay đổi bởi người khác.
* Tính sẵn sàng. Ví dụ: Đảm bảo máy chủ B luôn nhận được yêu cầu hợp lệ.

Ngoài ra còn 6 mục tiêu nữa:

* Xác thực thực thể. Ví dụ: A muốn xác nhận thông điệp được gửi từ B chắc chắn từ B gửi.
* Xác thực nguồn gốc của thông điệp. Ví dụ: A muốn chắc chắn thông điệp được cho là gửi đến từ B là từ B gửi và ngược lại.
* Timeliness – tính kịp thời: Ví dụ: ngăn chặn bên thứ 3 sao chép nội dung cuộc trò chuyện
* Chống từ chối: Ví dụ: A ko thể nào từ chối được thông điệp của mình đã gửi và đã nhận.
* Cấp phép: Ví dụ: một máy tính trên mạng có tài nguyên sẵn sàng cho một tập người dùng, nhưng ko phải tất cả người dùng mạng có thể truy cập được nguồn tài nguyên này.

Quyền truy cập: Ví dụ: giống như phân quyền trong DB

## Khái niệm mã hóa công khai? Sử dụng mã hóa công khai làm chữ ký số?

Mã hóa khóa công khai là một dạng mã hóa cho phép người sử dụng trao đổi các thông tin mật mà không cần phải trao đổi các khóa chung bí mật trước đó. Điều này được thực hiện bằng cách sử dụng một cặp khóa có quan hệ toán học với nhau là khóa công khai và khóa cá nhân (hay khóa bí mật).

Sử dụng mã công khai làm chữ ký số:

Bên A:

A muốn gửi thông điệp M và A sử dụng chữ ký số:

* A băm thông điệp M => 1 giá trị băm
* Giá trị băm dùng khoá private của A để mã hoá => chữ ký điện tử
* Đính kèm chữ ký điện tử với thông điệp M rồi gửi đến B.

Bên B:

* B gói tin của A gửi lấy ra thông điệp M và băm => giá trị băm (1)
* Lấy chữ ký mã hoá theo khoá public của A => giá trị băm (2)
* Nếu giá trị băm (1) bằng băm (2) thì chữ ký được xác thực và ngược lại

## Trình bày khái niệm firewall, cơ chế kiểm soát, ưu điểm và hạn chế?

Khái niệm Firewall: Firewall là cách thức để ngăn chặn truy cập tới mạng LAN hoặc cổng của mạng LAN. FireWall liên quan đến mục đích an ninh kiểm soát quyền truy cập.

**Cơ chế kiểm soát:**

+ Tất cả lưu lượng từ mạng bên trong ra bên ngoài hoặc ngược lại đều phải đi qua Firewall.

+ Chỉ có lưu lượng được phép, như được định danh bởi chính sách bảo mật cục bộ mới được phép đi qua Firewall

+ Bản thân Firewall phải có khả năng tránh được sự xâm nhập bất hợp pháp, và phải được thiết kế là một hệ thống tin cậy.

**Ưu điểm:**

+ Cung cấp 1 vị trí để giám sát các sự kiện liên quan đến bảo mật

+ Firewall định nghĩa một “choke point” cho sự quản lý những kết nối của mạng đến internet.

+ Ánh xạ địa chỉ mạng cục bộ thành địa chỉ trên Internet (NAT)

+ Firewall có thể phục vụ như một hệ nền cho IPSet.

**Hạn chế:**

+ Không thể bảo vệ khỏi các cuộc tấn công vượt qua tường lửa.

+ Không thể bảo vệ khỏi các cuộc tấn công nội bộ.

+ Không thể bảo vệ khỏi sự lây truyền của vi rút. Trong hầu hết các trường hợp, việc quét tất cả dữ liệu và tệp được chuyển vào là không thực tế hoặc không thể.

## Khái niệm về các phần mềm độc hại, phân loại?

* Phần mềm độc hại: là phần mềm xâm nhập vào hệ thống máy tính mà không có sự cho phép, không có sự hiểu biết hoặc chấp thuận của người dùng và sau đó thực hiện hành vi không mong muốn, gây hậu quả tiêu cực với máy tính chúng ta, thường có hại.
* Các loại phần mềm độc hại:
* Phân chia theo biến thể
* Phần mềm độc hại đơn hình (Oligomorphic malware): phần mềm độc hại này thay đổi mã nội bộ của nó dựa trên 1 số hữu hạn hình thái nhất định được xác định trước bất cứ khi nào nó được thực thi. Sau 1 chu kì biến đổi nhất định thì quay về hình thái ban đầu. Dễ bị phát hiện, bị loại bỏ bởi phần mềm diệt virus, scanner
* Phần mềm độc hại đa hình (Polymorphic malware): là mã phần mềm độc hại có thể thay đổi cấu trúc 1 cách toàn diện, thay đổi hình thức của nó, số lượng không giới hạn bất cứ khi nào được thực thi, gây khó khăn hơn trong quá trình phát hiện và tiêu diệt
* Phần mềm độc hại siêu hình (Metamorphic malware): là mã phần mềm độc hại có thể viết lại mã của chính nó sau mỗi "phiên bản", nó tập hợp lại với cùng logic và chức năng mà nó có trước đây nhưng thêm các yếu tố như code giả và các chức năng được sắp xếp lại để làm cho nó trông khác với các “phiên bản” trước.
* Chia theo các tiêu chí khác nhau
* Theo hình thức lây nhiễm: virus, worm, trojan, ..
* Theo hình thức che giấu thông tin: rootkit
* Thu thập dữ liệu người dùng: spiderware, ransomeware, adware
* Xóa dữ liệu: logic bomb
* Thay đổi hệ thống phần mềm: backdoor
* Phát động tấn công: zombie, botnet

## Phân loại firewall? Trình bày về cơ chế Package Filtering?

Phân loại Firewall:

Có 3 loại firewall phổ biến

+ Packet-Filtering Firewall: là loại firewall phổ biến nhất

+ Circuit-level gateway: làm việc ở cấp độ phiên của mô hình lớp mạng OSI

+ Application-level gateway: thường được sử dụng cùng với Packet-Filtering Firewall.

Cơ chế Packet-Filtering Firewall:

+ Firewall sẽ áp dụng một tập các quy tắc cho mỗi gói tin IP đi đến và sau đó quyết định sẽ chuyển tiếp hay loại bỏ gói tin đó đi.

+ Bộ lọc gói thường được thiết lập bằng danh sách quy tắc dựa vào các trường header IP và TCP.

+ Có 2 chính sách mặc định: discard và forward

## Nêu các hình thức tấn công? Liên hệ với các mục tiêu của an ninh mạng(Giống câu 1)

## Trình bày về Circuit‐level Gateway trong firewall ?

* Làm việc ở tầng vận chuyển của mô hình OSI, hoặc nó ở tầng “shim-layer” giữa tầng ứng dụng và tầng vận chuyển của ngăn TCP/IP
* Nó kiểm soát cơ chế bắt tay của TCP giữa các gói tin để xác định liệu 1 phiên yêu cầu có hợp lệ hay không.
* Là hệ thống tiêu chuẩn độc lập.
* Cổng thường chuyển tiếp các TCP segment từ 1 kết nối đến một kết nối khác mà ko cần xem xét nội dung của nó.
* Chức năng bảo mật bao gồm việc xác định những kết nối nào sẽ được cho phép
* Dùng trong trường hợp người quản lý hệ thống tin tưởng những người dùng nội bộ.

## Trình bày về các loại firewall topology?

Các loại firewall topology

* Simplest firewall topology: Đây là cấu trúc đơn giản nhất. Trong hầu hết các trường hợp, firewall là một packet filter firewall.
* Single Homed Host: Một Single Homed Host có một packet filter firewall với 1 cổng ứng dụng phía sau firewall. Cổng ứng dụng điều chỉnh quyền truy cập vào máy chủ thông tin. Nó cũng được sử dụng để điều chỉnh truy cập internet bởi các máy nội bộ. Máy chủ thông tin có thể được phép truy cập internet thông qua packet filter firewall, nhưng các thiết bị khác trên mạng nội bộ phải truy cập internet thông qua cổng ứng dụng.
* Dual Homed Bastion: Nó có 1 packet filter firewall với cổng ứng dụng phía sau firewall. Máy chủ lưu trữ điều chỉnh quyền truy cập vào máy chủ thông tin. Nó cũng được sử dụng để điều chỉnh truy cập vào internet bởi các máy tính cá nhân mạng nội bộ vì tất cả phải đi qua máy chủ mạng lưu trữ kép. Hệ thống này đảm bảo an ninh hơn vì ngay cả khi firewall bị xâm nhập, mạng nội bộ thì vẫn ẩn.
* DMZ: Có 1 packet filter firewall với cổng ứng dụng phía sau firewall. Có 1 packet filter firewall giữa cổng ứng dụng và mạng nội bộ. Khoảng cách giữa 2 packet filter firewall được gọi là DMZ. DMZ chứa tất cả các máy chủ có sẵn công khai của tổ chức. Đây là 1 cấu hình rất an toàn

## Khái niệm chữ ký số? Sử dụng mã hóa công khai thực hiện giao dịch có chữ ký số?

* Digital Signature là một kĩ thuật xác thực cho phép người chủ nội dung của một thông điệp được quyền đính kèm một đoạn dữ liệu số như là chữ kí đánh dấu của người chủ đối với nội dung đã tạo ra. Về cơ bản, chữ kí số sẽ được tạo ra bằng cách hash nội dung thông điệp sau đấy mã hoá chuỗi hash bằng khoá bí mật (private key) của người chủ nội dung.
* Sử dụng mã hóa công khai thực hiện giao dịch có chữ ký số:

Bên A:

A muốn gửi thông điệp M và A sử dụng chữ ký số:

* A băm thông điệp M => 1 giá trị băm
* Giá trị băm dùng khoá private của A để mã hoá => chữ ký điện tử
* Đính kèm chữ ký điện tử với thông điệp M rồi gửi đến B.

Bên B:

* B gói tin của A gửi lấy ra thông điệp M và băm => giá trị băm (1)
* Lấy chữ ký mã hoá theo khoá public của A => giá trị băm (2)
* Nếu giá trị băm (1) bằng băm (2) thì chữ ký được xác thực và ngược lại.

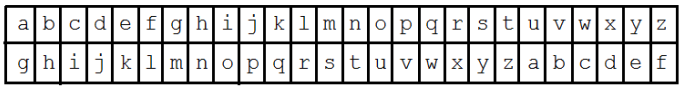
## Trình bày các bước để mã hóa theo thuật toán Caesar Cipher? Lấy ví dụ mã hóa một chuỗi ký tự tùy chọn để minh họa.

* B1: xoá tất các ký tự đặc biệt nằm ngoài bảng chữ cái (ví dụ: số, dấu chấm, dấu phẩy, dấu nháy...) và chuyển chữ hoa thành chữ thường.
* B2: Xác định 1 khoá là một số nằm trong khoảng từ 0 đến 25. Vẽ bảng thể hiện sự thay đổi của mỗi chữ cái.
* B3: Chuyển đổi bản rõ thành bản mã.
* Ví dụ:

Bản rõ : I am student

Khóa là 6

🡺 sự thay đổi của các chữ cái



🡺 Bản mã: o gs yzaktz

## Khi trao đổi thông điệp qua mạng tính toàn vẹn (integrity) được yêu cầu thế nào? Thực hiện yêu cầu toàn vẹn bằng giải pháp nào?

* Yêu cầu của tính toàn vẹn khi trao đổi thông điệp là đảm bảo thông điệp không bị sửa đổi bởi bất kỳ ai trong quá trình giao dịch.

Giải pháp:

* Hàm băm: là công cụ chính được sử dụng để cung cấp tính toàn vẹn

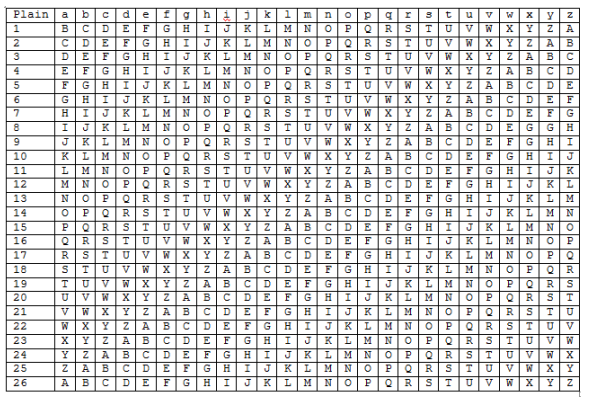
Có hai loại hàm băm:

+ Hàm băm không khóa: là những hàm băm không yêu cầu bất kì đầu vào nào ngoài thông điệp được băm

+ Hãm băm có khóa: là những hàm yêu cầu khóa cũng như thông điệp được băm

* Messeage Athentication Codes (MAC): là một đoạn thông tin ngắn được sử dụng để xác thực thư. Giá trị MAC bảo vệ cả tính toàn vẹn dữ liệu của tin nhắn cũng như tính xác thực của nó, bằng cách cho phép người xác minh (người cũng sở hữu khóa bí mật) phát hiện bất kỳ thay đổi nào đối với nội dung tin nhắn.

## Trình bày các bước để mã hóa theo thuật toán Vigenere Cipher? Lấy ví dụ mã hóa một chuỗi ký tự tùy chọn để minh họa.

* B1: xoá tất các ký tự đặc biệt nằm ngoài bảng chữ cái (ví dụ: số, dấu chấm, dấu phẩy, dấu nháy...) và chuyển chữ hoa thành chữ thường.
* B2: Chọn khoá là một từ.
* Bước 3 :đặt khóa liên tiếp vào bản rõ và đưa khóa về dạng chỉ số theo bảng chữ cái
* Bước 4: dịch chuyển kí tự của bản rõ đi 1 khoảng cách bằng chỉ số của kí tự trong khóa tương ứng
* 
* Ví dụ :
* Plaintext: toi la sinh vien khoa cntt
* Key: thuc hanh
* Bài làm:
* Key: t h u c h a n h t h u c h a n h
* Shifts: 19 7 20 2 7 0 13 7 19 7 20 2 7 0 13 7
* Plaintext: s i n h v i e n k h o a c n t t
* Ciphertext l p h j c i r u d o i c j n g a

## Nêu những hạn chế (nguy cơ bị bẻ khóa) của các phương pháp mã hóa kinh điển. Lấy ví dụ trong trường hợp Caesar Cipher và Vigenere Cipher

**- Phương pháp mã hóa kinh điển** là kỹ thuật đơn giản nên phương pháp mã hóa dữ liệu này có tính an toàn không cao. Việc giữ bí mật thuật toán cũng chỉ đảm bảo ở mức tương đối, nếu trường hợp nó bị rò rỉ ra ngoài hoặc kẻ xấu có thể lần mò giải ra thuật toán thì việc mã hóa trở nên vô nghĩa.

- Ví dụ: Caesar và Vigenere

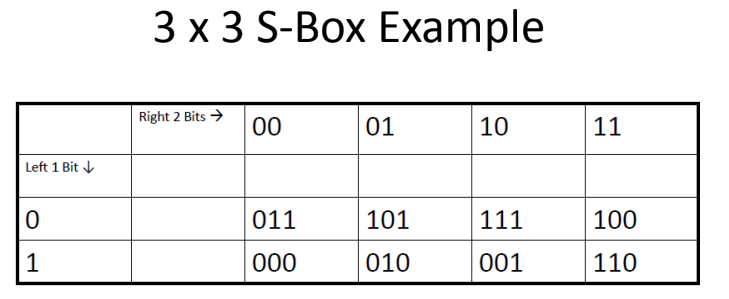
**Caesar Cipher**:rất dễ bị tấn công theo kiểu vét cạn vì chỉ có 25 trường hợp để ta biết được khóa,

phương pháp này không làm thay đổi tần suất xuất hiện của các kí tự nên có 1 cách tấn công nữa là dựa vào tần suất xuất hiện của các kí tự để đoán ,ví dụ trong 1 bảng thống kê kí tự e xuất hiện trong các văn bản là 12.7%...vậy nếu thống kê trên bản mã mà có 1 kí tự nào đấy xấp xỉ 12.7% thì hoàn toàn có thể có cơ sở để đoán kí tự đấy là biến thể của kí tự e.

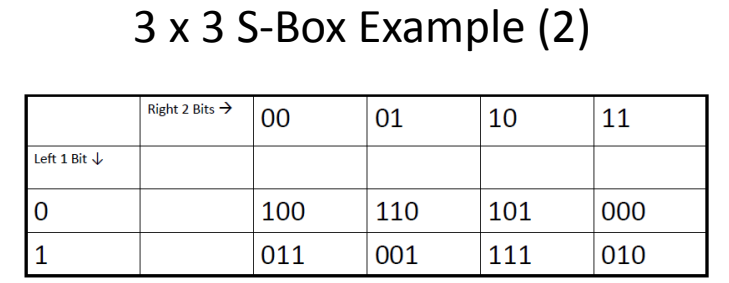
**Vigenere Cipher**  : Nếu khóa là một từ hoặc cụm từ có nghĩa thì độ mạnh sẽ giảm đi vì khóa dễ đoán hơn. Có thể sử dụng các kỹ thuật để ước tính độ dài của khóa .

## Trên cơ sở phân tích các hạn chế của các thuật toán mã hóa cổ điển (Caesar Cipher, Vigeniar Cipher), các chuẩn mã hóa tiên tiến được thiết kế như thế nào? Trình bày các phép biến đổi cơ bản hay được sử dụng trong các chuẩn mã hóa tiên tiến.

* Các chuẩn mã hóa tiên tiến được thiết kế xoay quanh việc sử dụng “khóa bí mật” để mã hóa dữ liệu, áp dụng kích thước khóa lớn hơn, nhiều khóa hơn, đồng thời thực hiện mã hóa nhiều vòng để làm tăng tính bảo mật và đem lại độ an toàn.
* Phép biến đổi cơ bản hay được sử dụng trong các chuẩn mã hóa tiên tiến:
* Substitution(thay thế): lấy m bit đầu vào và biến đổi chúng thành n bit đầu ra, n không nhất thiết bằng m

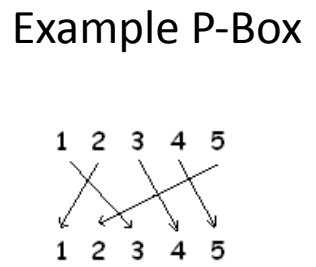


Đây là ví dụ về 3x3 S-box lấy 3 bit đầu vào và đầu ra là 3 bit



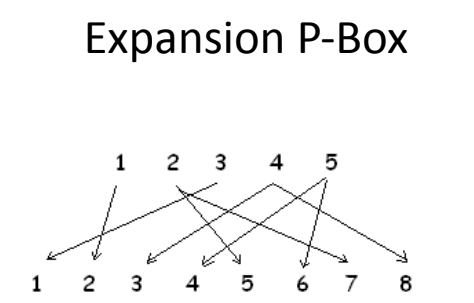
Đây là hộp giải mã của hộp 3x3 S-box trước

* Transposition(hoán vị):gọi là p-box chia đầu vào thành các khối n bit và hoán vị từng khối thì được khối đầu ra



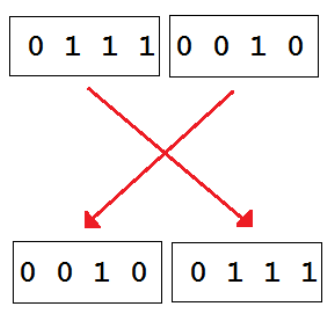
Trên là đổi vị trí 5 bit

Ta có thể mở rộng từ 5 bit biến đổi thành 8 bit

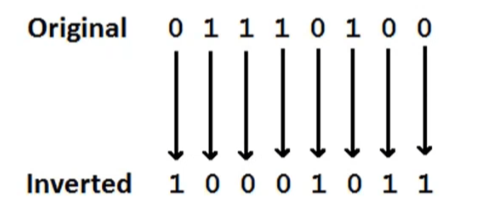


Đây là 1 ví dụ về hoán vị nhưng có mở rộng

* Swap (phép đảo) 1 tập hợp số chẵn các bit được chia thành 2 nửa và các nửa hoán đổi cho nhau



* Bit inversion(phép đảo bit) đảo bit 0 thành 1 ,1 thành 0



* Circular Shift(phép dịch bit):các bit được dịch trái hoặc phải ,không bit nào bi thay đổi nhưng vị trí sẽ thay đổi
* XOR Operation(phép xor bit): lấy hai dãy bit có cùng độ dài và thực hiện phép toán logic bao hàm **XOR** trên mỗi cặp bit tương ứng. Kết quả ở mỗi vị trí là 1 chỉ khi bit đầu tiên là 1 hoặc nếu chỉ khi bit thứ hai là 1, nhưng sẽ là 0 nếu cả hai là 0 hoặc cả hai là 1.

## Nêu các hạn chế của firewall? Các cơ chế thiết lập firewall?

- Các hạn chế của firewall:

+ Firewall ko thể chống lại các cuộc tấn công ko đi qua nó.

+ Firewall ko thể chống lại các nguy cơ tấn công từ chính bên trong mạng nội bộ mà nó bảo vệ

+ Firewall ko thể bảo vệ chống lại việc chuyển các chương trình hoặc file có virus đi qua nó.

- Các cơ chế thiết lập firewall:

**+** Tất cả truy cập lưu lượng đến hoặc đi, internet đều phải đi qua firewall

**+** Chỉ lưu lượng được ủy quyền, như được xác định bởi chính sách bảo mật cục bộ mới có thể đi qua firewall.

**+** Bản thân firewall không bị xâm nhập. Nó là một hệ thống đáng tin cậy.

## Trình bày cách thiết lập một bộ khóa trong thuật toán RSA. Sử dụng bộ khóa đó để mã hóa và giải mã một ký tự tùy chọn

Cách thiết lập một bộ khóa trong thuật toán RSA:

1. Chọn 2 số nguyên tố lớn khác nhau p, q thỏa mãn điều kiện (lựa chọn ngẫu nhiên và độc lập)

2. Tính tích của p, q: n = p\*q;

3. Tính giá trị hàm số Euler của n: .

4. Tính1 số tự nhiên e sao cho 1<e< và là số nguyên tố cùng nhau với .

5. Chọn số nguyên d, sao cho và gcd (d, e) ≡ 1 (mod )

6. Khóa công khai bao gồm: n và e. Khóa bí mật gồm: n và d.

- Ứng dụng:

1) Chọn p = 11 và q = 3, do đó N = p\*q = 33 ( = 32 < 33 < 64 = )

2) n = (p-1) (q-1) = 20

3) Chọn e = 3 nguyên tố cùng nhau với n

4) Tính nghịch đảo của e trong phép modulo n được d = 7 (3x7 = 21)

5) Khóa công khai KU = (e, N) = (3, 33). Khóa bí mật KR = (d, N) = (7, 33)

6) Mã hóa bản rõ M = 15: C = mod N = 153 mod 33 = 9

7) Giải mã bản mã C = 9: M = mod N = 97 mod 33 = 15

## Nêu những hiểu biết của bạn về hàm băm (Hash function) và sử dụng hàm băm trong đảm bảo tính toàn vẹn của thông điệp?

\* Hiểu biết về hàm băm:

- Một hàm băm thực hiện trên 1 thông điệp để tạo ra 1 giá trị băm.

- Thông điệp được băm được thay đổi nhỏ thì sẽ đưa ra 1 giá trị băm mới khác hoàn toàn nhau.

- Hàm băm thì băm trên các thông điệp khác nhau thì rất khó tạo ra giá trị băm giống nhau.

- Rất khó từ giá trị băm để suy ra được thông điệp gốc.

\* Sử dụng hàm băm trong đảm bảo tính toàn vẹn

- A gửi thông điệp M cho B thì A sẽ băm M để được 1 giá trị băm sau đó gán cùng với thông điệp M gửi cho B

- Bên B nhận được thông điệp lấy ra được thông điệp M sau đó băm M và so sánh 2 giá trị băm, nếu trùng khớp thì dữ liệu được toàn vẹn.

## Nêu kiến trúc các mô hình cài đặt firewall?

- **Thứ nhất: Kiến trúc Dual-homed Host**

Được xây dựng dựa trên máy tính dual-home host. Một máy tính được gọi là dual-homed host nếu có ít nhất 2 network interface, có nghĩa là máy đó có gắn 2 card giao tiếp với 2 mạng khác nhau , do đó máy tính này đóng vai trò là router mềm, Kiến trúc dual-home host rất đơn giản, máy dual-home host ở giữa, một bên được nối với internet và bên còn lại nối với mạng nội bộ. Kiến trúc này có các đặc điểm sau:

+ Phải disable chức năng routing của dual-homed host để cấm hoàn toàn thông IP từ ngoài vào

+ Các hệ thống bên ngoài và bên trong dual-home host không liên lạc trực tiếp được với nhau.

+ Dual homed host cung cấp dịch vụ thông qua poxy server hoặc login trực tiếp vào dual –home host.

- **Thứ hai: Kiến trúc Screened Host**

Trong kiến trúc này chức năng bảo mật chính được cung cấp bởi chức năng packet filtering tại screening router. Packet filtering trên screening router được setup sao cho basition host là máy duy nhất trong internal network mà các host trên internet có thể mở kết nối đến Packet filtering cũng cho phép bastion host mở các kết nối ra bên ngoài. Các packet filtering thực hiện công việc sau:

+ Cho phép các interal hosts mở kết nối đến các host trên internet đối với một số dịch vụ được phép

+ Cấm tất cả kết nối từ các interal hosts

- **Thứ ba: Kiến trúc Screened Subnet Host**

Thêm một perimeter network để cô lập interal network với internet.Như vật dù hacker đã tấn công được vào bastion host vẫn còn một rào chắn nữa phải vượt qua interior router. Các lưu thông trong internet network được bảo vệ an toàn cho dù bastion đã bị “ chiếm”. Các dịch vụ nào ít tin cậy và có khả năng dễ bị tấn công thì nên để ở perimeter network.Bastion host là điệm liên lạc cho các kết nối từ ngoài vào :SMTP, FTP,DNS . Còn đối với viêc truy cập các dịch vụ từ internal client đến các server trên internet thì được điều khiển như sau:

Setup packet filtering trên các exterior và interior router để cho phép internal client truy cập các servers bên ngoài 1 cách trực tiếp.

Setup poxy server tren bastion host để cho phép internal clients truy cập các server bên ngoài 1 cách gián tiếp.

## Trình bày về SQL Injection và cách khắc phục

**- SQL Injection** là một kỹ thuật lợi dụng những lỗ hổng về câu truy vấn của các ứng dụng. Được thực hiện bằng cách chèn thêm một đoạn [SQL](https://topdev.vn/blog/sql-la-gi/) để làm sai lệnh đi câu truy vấn ban đầu, từ đó có thể khai thác dữ liệu từ database. **SQL injection** có thể cho phép những kẻ tấn công thực hiện các thao tác như một người quản trị web, trên cơ sở dữ liệu của ứng dụng.

- Cách khắc phục:

+ Cách 1: Vô hiệu hóa các ký tự đặc biệt: loại bỏ các ký tự đặc biệt như dấu ngoặc kép, dấu nháy bằng đặt trước chúng dấu gạch chéo ngược. Hoặc sử dụng phương thức get\_magic\_quotes\_gpc() hoặc sử dụng phương thức real\_escape\_string() khi gọi đến MySQL.

+ Cách 2: Tham số hóa cho câu truy vấn: Không cho phép dữ liệu do người dùng gửi được hiểu là các câu lệnh MySQL. Câu lệnh truy vấn có thể sử dụng tham số ‘?’ như 1 từ thay thế cho dữ liệu.

# Bài tập

## Sử dụng Java Cryptography Architecture (JCA), viết class SymmetricCryp để thực hiện các chức năng mã hóa và giải mã đối xứng theo thuật toán AES gồm các hàm

// Các lớp và thông tin sử dụng

// thuật toán

**private** **final** String algorithms = "AES";

// khóa

**private** SecretKey secretKey;

// bộ sinh khóa theo thuật toán

**private** KeyGenerator keyGen;

// bộ mã

**private** Cipher cipher;

// Chuỗi tạo khóa

**private** String key = "0123456789101112";

// key spec theo chuỗi cho trước

**private** SecretKeySpec secretKeySpec;

### Sinh khóa đối xứng

**public** SymmetricCryptor() **throws** Exception {

// sinh khóa từ generator với thuật toán được chọn

keyGen = KeyGenerator.*getInstance*(algorithms);

secretKey = keyGen.generateKey();

}

**public** SecretKey getSecretKey() {

**return** secretKey;

}

### Tạo khóa đối xứng từ một chuỗi cho trước

**public** SymmetricCryptor() **throws** Exception {

// tạo key spec chuỗi đã cài đặt từ trước

secretKeySpec = **new** SecretKeySpec(key.getBytes(), algorithms);

}

**public** SecretKeySpec getSecretKeySpec() {

**return** secretKeySpec;

}

### Mã hóa thông điệp có 2 tham số: Tham số 1 là chuỗi cần mã hóa, tham số thứ 2 là khóa được tạo ở 1.1

// msg : chuỗi cần mã hóa

// key : khóa được tạo từ 1

// cách này sử dụng khóa cung cấp sẵn từ SecretKey

**public** String encryptText(String msg, SecretKey key) **throws** Exception {

cipher = Cipher.*getInstance*(algorithms);

cipher.init(Cipher.***ENCRYPT\_MODE***, key);

**return** Base64.*getEncoder*().encodeToString(cipher.doFinal(msg.getBytes("UTF-8")));

}

### Mã hóa thông điệp có 2 tham số: Tham số 1 là chuỗi cần mã hóa tham số 2 là khóa được tạo ở 1.2

// msg : chuỗi cần mã hóa

// key : khóa được tạo từ 1

// cách này sử dụng khóa sinh ra từ lớp SecretKeySpec với chuỗi xác định trong 1.1

**public** String encryptText1(String msg, SecretKeySpec key) **throws** Exception {

cipher = Cipher.*getInstance*(algorithms);

cipher.init(Cipher.***ENCRYPT\_MODE***, key);

**return** Base64.*getEncoder*().encodeToString(cipher.doFinal(msg.getBytes("UTF-8")));

}

### Giải mã có 2 tham số: Tham số 1 là bản mã, tham số thứ 2 là khóa bí mật được tạo ở 1.1

**public** String decryptText(String msg, SecretKey key) **throws** Exception {

cipher = Cipher.*getInstance*(algorithms);

cipher.init(Cipher.***DECRYPT\_MODE***, key);

**return** **new** String(cipher.doFinal(Base64.*getDecoder*().decode(msg)), "UTF-8");

}

### Giải mã có 2 tham số: Tham số 1 là bản mã, tham số thứ 2 là khóa bí mật được tạo ở 1.2

**public** String decryptText1(String msg, SecretKeySpec key) **throws** Exception {

cipher = Cipher.*getInstance*(algorithms);

cipher.init(Cipher.***DECRYPT\_MODE***, key);

**return** **new** String(cipher.doFinal(Base64.*getDecoder*().decode(msg)), "UTF-8");

}

### Sử dụng class SymetricCryp để mã hóa một chuỗi cho trước hoặc giải mã ra bản rõ từ bản mã cho trước.

**public** **class** SymmetricCrt {

**public** **static** **void** main(String[] args) **throws** Exception {

SymmetricCryptor Sc = **new** SymmetricCryptor();

String msg = "Nguyễn Văn An";

String en\_msg = Sc.encryptText(msg, Sc.getSecretKey());

System.***out***.println("Plain text: " + msg);

System.***out***.println("Encrypted text: " + en\_msg);

String de\_msg = Sc.decryptText(en\_msg, Sc.getSecretKey());

System.***out***.println("Decrypted text: " + de\_msg);

}

}

## Sử dụng Java Cryptography Architecture (JCA) để viết các hàm băm thông điệp cho trước theo thuật toán:

### MD5

**public** **class** messageauthentication {

**public** **static** **void** main(String[] args) **throws** Exception {

String msg = "Nguyễn Văn An";

// khởi tạo bộ tạo bản băm với tham số vào là tên thuật toán

MessageDigest md = MessageDigest.*getInstance*("MD5");

// thiết lập đầu vào cho bản băm là chuỗi byte

md.update(msg.getBytes("UTF-8"));

//gọi hàm md.digest để thu được bản băm là chuỗi byte đầu ra

**byte** byteData[] = md.digest();

// hiển thị chuỗi byte đầu ra dạng mã Hexa

StringBuffer hexString = **new** StringBuffer();

**for** (**int** i = 0; i < byteData.length; i++) {

String hex = Integer.*toHexString*(0xff & byteData[i]);

**if** (hex.length() == 1)

hexString.append('0');

hexString.append(hex);

}

System.***out***.println("Hex format : " + hexString.toString());

}

}

### SHA1

**public** **class** MessageAuthentication {

**public** **static** **void** main(String[] args) **throws** Exception {

MessageDigest md = MessageDigest.*getInstance*("SHA-1");

String msg = "Nguyễn Văn An";

md.update(msg.getBytes("UTF-8"));

**byte**[] hashbytes = md.digest();

// chuyển sang chuỗi hexa để hiển thị

StringBuffer buff = **new** StringBuffer();

**int** i;

**for** (i = 0; i < hashbytes.length; i++) {

String s = Integer.*toHexString*(hashbytes[i] & 0xff);

**if** (s.length() == 1)

s += "0";

buff.append(s);

}

System.***out***.println("Source message: " + msg);

System.***out***.println("Hash code: " + buff);

}

}

### SHA-256

**public** **class** MessageAuthentication {

**public** **static** **void** main(String[] args) **throws** Exception {

MessageDigest md = MessageDigest.*getInstance*("SHA-256");

String msg = "Nguyễn Văn An";

md.update(msg.getBytes("UTF-8"));

**byte**[] hashbytes = md.digest();

// chuyển sang chuỗi hexa để hiển thị

StringBuffer buff = **new** StringBuffer();

**int** i;

**for** (i = 0; i < hashbytes.length; i++) {

String s = Integer.*toHexString*(hashbytes[i] & 0xff);

**if** (s.length() == 1)

s += "0";

buff.append(s);

}

System.***out***.println("Source message: " + msg);

System.***out***.println("Hash code: " + buff);

}

}

## Sử dụng Java Crypto Architecture (JCA), viết chương trình mã hóa và giải mã sử dụng mã hóa công khai RSA: Viết class RSAKeyGen để sinh bộ khóa công khai gồm các hàm

### Sinh bộ khóa theo thuật toán RSA

**private** KeyPairGenerator keyGen; // đối tượng keyPairGenerator là bộ sinh khóa

**private** KeyPair keypair; // đối tượng keyPair lưu giữ cặp khóa

**private** PublicKey publicKey; // đối tượng public key lưu giữ khóa công khai

**private** PrivateKey privateKey; // đôi tượng private key lưu giữ khóa bí mật

**public** KeyPairGen(**int** keylength) **throws** NoSuchAlgorithmException {

// khởi tạo bộ tạo khóa keyPairGenerator với ttoan RSA

keyGen = KeyPairGenerator.*getInstance*("RSA");

// khởi tạo bộ sinh khóa với tham số đầu vào là byte

keyGen.initialize(keylength);

// sinh khóa

keypair = keyGen.genKeyPair();

// lấy khóa công khai

publicKey = keypair.getPublic();

// lấy khóa bí mật

privateKey = keypair.getPrivate();

}

**public** PublicKey getPublicKey() {

**return** publicKey;

}

**public** PrivateKey getPrivateKey() {

**return** privateKey;

}

### Lưu bộ khóa ra file

// ghi khóa ra file

**public** **void** writeToFile(String path, **byte**[] key) **throws** FileNotFoundException, IOException {

File f = **new** File(path);

// tạo thư mục lưu trữ 2 khóa

f.getParentFile().mkdir();

FileOutputStream fos = **new** FileOutputStream(f);

fos.write(key);

fos.flush();

fos.close();

}

Viết class RSACryp để sử dụng bộ khóa đã lưu trong file để mã hóa và giải mã thông điệp theo thuật toán RSA

**public** **class** RSACryp {

**private** Cipher cipher;

### Hàm đọc file để nạp khóa bí mật, tham số truyền vào là tên file

**public** PrivateKey getPrivateKey(String filename)

**throws** IOException, NoSuchAlgorithmException, InvalidKeySpecException {

**byte**[] keybytes = Files.*readAllBytes*(**new** File(filename).toPath());

PKCS8EncodedKeySpec spec = **new** PKCS8EncodedKeySpec(keybytes); // lấy private key từ file (theo bytes)

KeyFactory kf = KeyFactory.*getInstance*("RSA");

**return** kf.generatePrivate(spec);

}

### Hàm đọc file để nạp khóa công khai, tham số truyền vào là tên file

**public** PublicKey getPublicKey(String filename)

**throws** IOException, NoSuchAlgorithmException, InvalidKeySpecException {

**byte**[] keybytes = Files.*readAllBytes*(**new** File(filename).toPath());

X509EncodedKeySpec spec = **new** X509EncodedKeySpec(keybytes); // lấy pkey từ file (theo bytes)

KeyFactory kf = KeyFactory.*getInstance*("RSA");

**return** kf.generatePublic(spec);

}

### Hàm mã hóa một thông điệp cho trước bằng khóa bí mật, tham số truyền vào gồm tham số thứ nhất là thông điệp cần mã hóa, tham số thứ 2 là khóa bí mật

**public** String encryptText(String msg, PrivateKey key) **throws** Exception { // chữ ký số

cipher = Cipher.*getInstance*("RSA");

cipher.init(Cipher.***ENCRYPT\_MODE***, key);

**return** Base64.*getEncoder*().encodeToString(cipher.doFinal(msg.getBytes("UTF-8")));

}

### Hàm giải mã thông điệp bằng khóa công khai, tham số truyền vào gồm tham số thứ nhất là bản mã cần giải mã, tham số thứ 2 là khóa công khai.

**public** String decryptText(String msg, PublicKey key) **throws** Exception {

cipher = Cipher.*getInstance*("RSA");

cipher.init(Cipher.***DECRYPT\_MODE***, key);

**return** **new** String(cipher.doFinal(Base64.*getDecoder*().decode(msg)), "UTF-8");

}

### Sử dụng lớp RSACryp để mã hóa một chuỗi và băm chuỗi theo một thuật toán băm cho trước rồi lưu cả bản mã và bản băm vào một file

**public** **void** writeToFile(String path, String data) **throws** FileNotFoundException, IOException {

File f = **new** File(path);

// lưu file với string cho trước

f.getParentFile().mkdir();

FileOutputStream fos = **new** FileOutputStream(f);

fos.write(data.getBytes());

fos.flush();

fos.close();

}

**public** **class** AsymmetricDemo {

**public** **static** **void** main(String[] args) **throws** Exception {

RSACryp rsaCryp = **new** RSACryp();

**try** {

MessageDigest digest = MessageDigest.*getInstance*("SHA-1");

KeyPairGen rsaKeyGen = **new** KeyPairGen(1024);

rsaKeyGen.writeToFile("D:/Keys/PrivateKey", rsaKeyGen.getPrivateKey().getEncoded());

rsaKeyGen.writeToFile("D:/Keys/PublicKey", rsaKeyGen.getPublicKey().getEncoded());

String msg = "Nguyen Van An";

String encrypted\_msg = rsaCryp.encryptText(msg, rsaCryp.getPrivateKey("D:/Keys/PrivateKey"));

**byte**[] firstHash = digest.digest(msg.getBytes("UTF-8"));

String hashCiper = encrypted\_msg + ";" + Base64.*getEncoder*().encodeToString(firstHash);

rsaKeyGen.writeToFile("D:/result.txt", hashCiper);

} **catch** (Exception e) {

e.printStackTrace();

}

### Đọc file chứa bản mã và bản băm của một thông điệp, sử dụng lớp RSACryp để giải mã ra thông điệp gốc và băm lại bằng thuật toán băm cho trước. Sau đó đem so sánh hai bản băm để xác nhận tính toàn vẹn

**public** String readFile(String path) {

**try** {

// đọc file và chuyển sang String

**byte**[] rs = Files.*readAllBytes*(Paths.*get*(path));

**return** **new** String(rs, StandardCharsets.***UTF\_8***);

} **catch** (IOException e) {

e.printStackTrace();

}

**return** **null**;

}

String dataRead = rsaKeyGen.readFile("D:/result.txt");

String result[] = dataRead.split(";");

String decrypted\_msg = rsaCryp.decryptText(result[0], rsaCryp.getPublicKey("D:/Keys/PublicKey"));

**byte**[] hashAgain = digest.digest(decrypted\_msg.getBytes("UTF-8"));

String hashAgainString = Base64.*getEncoder*().encodeToString(hashAgain);

System.***out***.println("Plain Text:" + msg);

System.***out***.println("Encrypt Text: " + encrypted\_msg);

System.***out***.println("MD5 hash: " + **new** String(firstHash));

System.***out***.println("MD5 hash: " + result[1]);

System.***out***.println("Decrypt Text: " + decrypted\_msg);

System.***out***.println("Hash Again: " + hashAgainString);

**if**(hashAgainString.equals(result[1])){

System.***out***.println("Same"); // toàn vẹn

} **else** {

System.***out***.println("Not same"); // không toàn vẹn

}

} **catch** (Exception e) {

e.printStackTrace();

}

## Lập trình mô tả lỗi SQL Injection và cách khắc phục

* Mô tả lỗi:
* <?php
* if (isset($\_POST['user']) && isset($\_POST['pass'])) {
* $user = $\_POST['user'];
* $pass = $\_POST['pass'];
* if ($user != '' && $pass != '') {
* $host = 'localhost';
* $login = 'root';
* $passlog =;
* $dbname = 'test';
* $port = 3306;
* $conn = new *mysqli*($host, $login, $passlog, $dbname, $port);
* if (!$conn) die($conn->error\_connect);
* $sql = "select \* from account where user = '" . $user . "' and pass = '" . $pass . "' ";
* echo $sql;
* $result = $conn->query($sql);
* if (!$result) die($conn->error);
* if (count(mysqli\_fetch\_all($result)) > 0)
* echo "<h1>Login successfully</h1>";
* else  echo "<h1>Login Fail</h1>";
* } else {
* echo "<h1>username and password are wrong</h1>";
* }
* }
* ?>
* Trong đoạn code trên nếu ô user và pass ta nhập là ‘ or ‘1’=’1 thì biến $sql = Select \* from account where user = ‘’ or ‘1’=’1’ and pass = ‘’ or ‘1’=’1’ .Điều này là luôn đúng và hệ thống sẽ luôn trả ra Login successfully.
* **Cách khắc phục**
* Cách 1: Vô hiệu hóa các ký tự đặc biệt: loại bỏ các ký tự đặc biệt như dấu ngoặc kép, dấu nháy bằng đặt trước chúng dấu gạch chéo ngược. Hoặc sử dụng phương thức get\_magic\_quotes\_gpc() hoặc sử dụng phương thức real\_escape\_string() khi gọi đến MySQL.

function mysql\_fix\_string($conn, $string)

{

if (get\_magic\_quotes\_gpc()) {

$string = stripcslashes($string);

return $conn->real\_escape\_string($string);

}

}

?>

<?php

$user = $\_POST['user'];

$pass = $\_POST['pass'];

$host = 'localhost';

$login = 'root';

$passlog = '';

$dbname = 'php\_sql\_injection\_test';

$port = 3306;

$conn = new mysqli($host, $login, $passlog, $dbname, $port);

if (!$conn)

die($conn->error\_connect);

$user = mysql\_fix\_string($conn, $user);

$pass = mysql\_fix\_string($conn, $pass);

$sql = "SELECT \* FROM `account` where un='" . $user . "' AND pw= '" . $pass . "'";

$result = $conn->query($sql);

if (!$result)

die($conn->error);

if (count(mysqli\_fetch\_all($result)) > 0)

echo "<h1>Login successfully! ! !</h1>";

else

echo "<h1>Login failed!</h1>";

?>

* Tham số hóa cho câu truy vấn: Không cho phép dữ liệu do người dùng gửi được hiểu là các câu lệnh MySQL. Câu lệnh truy vấn có thể sử dụng tham số ‘?’ như 1 từ thay thế cho dữ liệu (kí hiệu giữ chỗ - placeholder).

<?php

$user = $\_POST['user'];

$pass = $\_POST['pass'];

$host = 'localhost';

$login = 'root';

$passlog = '';

$dbname = 'php\_sql\_injection\_test';

$port = 3306;

$conn = new mysqli($host, $login, $passlog, $dbname, $port);

if (!$conn)

die($conn->error\_connect);

$sql = "SELECT \* FROM `account` WHERE un = ? AND pw = ?";

            $stmt = $conn->prepare($sql);

            $stmt->bind\_param("ss", $user, $pass);

            if ($stmt->execute()) {

                echo $stmt->fullQuery;

                $result = $stmt->get\_result();

                $num\_rows = $result->num\_rows;

            }

            if ($num\_rows > 0) {

                echo "<h1>Login successfully! ! !</h1>";

            } else {

                echo "<h1>Login failed!</h1>";

            }

?>