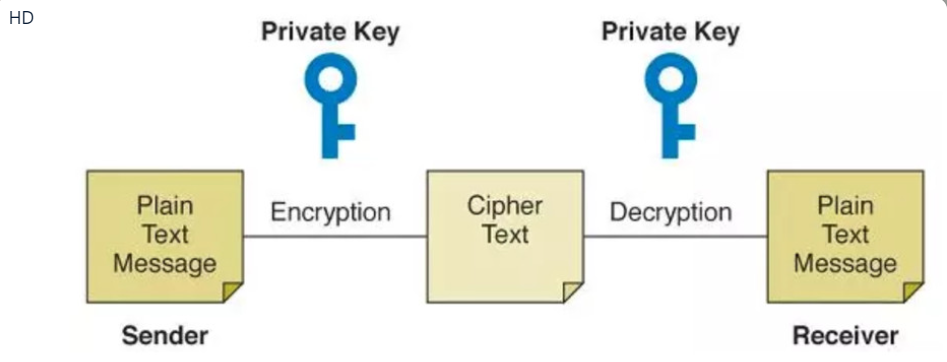
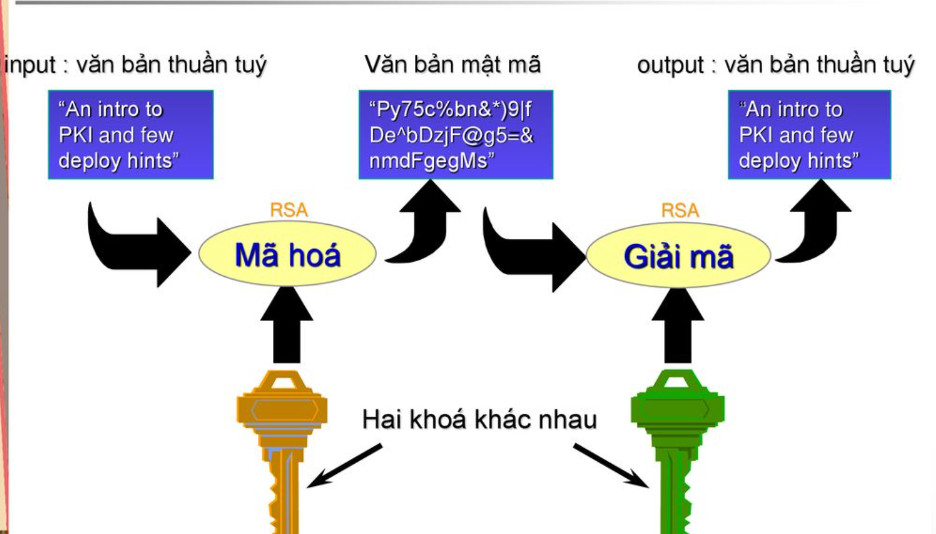
**SEM 3 – NSC (Network Security and Cryptography) Exam Questions**

**Câu 1 (5 points). Mã hoá đối xứng và bất đối xứng**

* 1. **Khái niệm**
* Mã hoá đối xứng (Symmetric encryption) là một phương pháp mã hoá dữ liệu trong đó cùng một khóa được sử dụng để mã hoá và giải mã thông tin. Người gửi và người nhận cùng sử dụng một khóa chung để mã hoá và giải mã dữ liệu. Mã hoá đối xứng nhanh và hiệu quả, nhưng đòi hỏi sự an toàn và bảo mật của khóa chung.
* Mã hoá bất đối xứng (Asymmetric encryption) là một phương pháp mã hoá dữ liệu trong đó cần hai khóa riêng biệt: một khóa công khai (public key) và một khóa riêng tư (private key). Khóa công khai được chia sẻ với mọi người, trong khi khóa riêng tư chỉ được giữ bí mật bởi người nhận. Dữ liệu được mã hoá bằng khóa công khai và chỉ có thể giải mã bằng khóa riêng tư tương ứng. Mã hoá bất đối xứng an toàn hơn mã hoá đối xứng, nhưng tốn nhiều tài nguyên tính toán hơn.
  1. **Biểu đồ mã hoá**
* Mã hoá đối xứng:



* Mã hoá bất đối xứng:



* 1. **So sánh**

+ Mã hóa đối xứng:

* Khóa: Mã hóa đối xứng sử dụng cùng một khóa để thực hiện cả quá trình mã hóa và giải mã. Khóa này cần được chia sẻ giữa người gửi và người nhận, điều này đòi hỏi quản lý khóa an toàn và hiệu quả.
* Tính đối xứng: Quá trình mã hóa và giải mã đều sử dụng cùng một khóa, tạo ra tính đối xứng trong cách thực hiện. Điều này có nghĩa là nếu ai đó biết khóa mã hóa, họ cũng có khả năng giải mã dữ liệu.
* – Tốc độ: Mã hóa đối xứng thường nhanh hơn mã hóa bất đối xứng vì chỉ cần thực hiện một loạt các phép biến đổi đơn giản.

+ Mã hóa bất đối xứng:

* Khóa: Mã hóa bất đối xứng sử dụng một cặp khóa, gồm khóa công khai và khóa riêng tư. Khóa công khai được chia sẻ công khai và dùng để mã hóa dữ liệu, trong khi khóa riêng tư được giữ bí mật và dùng để giải mã dữ liệu. Không cần chia sẻ khóa riêng tư.
* Tính bất đối xứng: Khóa công khai và khóa riêng tư đóng vai trò không đối xứng, điều này có nghĩa là dữ liệu mã hóa bằng khóa công khai chỉ có thể được giải mã bằng khóa riêng tư, và ngược lại.
* Độ bảo mật: Mã hóa bất đối xứng thường đảm bảo độ bảo mật cao hơn do không cần chia sẻ khóa riêng tư. Người gửi chỉ cần biết khóa công khai của người nhận để mã hóa dữ liệu, và người nhận giữ khóa riêng tư để giải mã.

**Tóm lại:** Sự khác biệt cơ bản giữa mã hóa đối xứng và mã hóa bất đối xứng là trong việc sử dụng khóa và cách thức thực hiện quá trình mã hóa và giải mã. Mã hóa đối xứng sử dụng cùng một khóa cho cả hai quá trình, trong khi mã hóa bất đối xứng sử dụng cặp khóa khác nhau cho việc mã hóa và giải mã. Mã hóa bất đối xứng thường đảm bảo tính bảo mật cao hơn, trong khi mã hóa đối xứng thường nhanh hơn trong việc thực hiện quá trình mã hóa và giải mã.

* 1. **Viết chương trình mã hóa và giải mã bất đối xứng bằng ngôn ngữ C#.**

**Câu 2(3 points). Lỗ hổng bảo mật**

**2.1 Khái niệm**

-Lỗ hổng bảo mật (Security vulnerability) là một điểm yếu hoặc một sự cố trong hệ thống, ứng dụng hoặc thiết bị mà có thể bị tấn công hoặc lợi dụng để xâm nhập, truy cập trái phép, thay đổi hoặc đánh cắp thông tin, hoặc gây hại đến hệ thống. Lỗ hổng bảo mật có thể là do lỗi trong thiết kế, triển khai, cấu hình, phần mềm, phần cứng hoặc cả con người.

**2.2 SQL Injection và cách phòng tránh**

**- 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 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 phòng tránh:

* Sử dụng Prepared Statements: Sử dụng Prepared Statements (còn được gọi là parameterized queries) để tạo các truy vấn SQL. Prepared Statements sẽ tự động xử lý việc truyền các tham số vào truy vấn, giúp ngăn chặn việc nhập liệu người dùng trực tiếp vào truy vấn SQL.
* Sử dụng Stored Procedures: Sử dụng Stored Procedures để thực hiện các truy vấn SQL. Stored Procedures được lưu trữ và quản lý trên server database, giúp ngăn chặn việc nhập liệu người dùng trực tiếp vào truy vấn SQL.
* Kiểm tra và xử lý đầu vào: Kiểm tra và xử lý đầu vào từ người dùng trước khi sử dụng chúng trong truy vấn SQL. Bạn có thể sử dụng các hàm như mysqli\_real\_escape\_string hoặc PDO::quote để thoát các ký tự đặc biệt trong đầu vào.
* Hạn chế quyền truy cập: Cấp quyền truy cập database cho người dùng với các quyền tối thiểu cần thiết. Điều này giúp giới hạn khả năng thực hiện các truy vấn SQL độc hại.
* Cập nhật hệ thống: Đảm bảo rằng hệ thống của bạn được cập nhật đầy đủ và bảo mật. Cài đặt các bản vá bảo mật và các bản cập nhật mới nhất để ngăn chặn các lỗ hổng bảo mật có thể được khai thác.
* Sử dụng Firewall và WAF: Cài đặt Firewall và Web Application Firewall (WAF) để ngăn chặn các cuộc tấn công từ xa và kiểm soát lưu lượng truy cập đến ứng dụng của bạn.
* Kiểm tra và giám sát: Thực hiện kiểm tra bảo mật định kỳ và giám sát các hoạt động của hệ thống để phát hiện và ngăn chặn các cuộc tấn công SQL Injection.

**Câu 3(3 points): Xác thực**

**Sử dụng phương pháp xác thực mật khẩu, hãy tìm hiểu một chương trình bằng ngôn ngữ C# mã hóa (SHA-256) chuỗi mật khẩu của người dùng**