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

**Question 1: About asymmetric symmetric encryption**

**1.1 Concept:**

**Symmetric encryption** (symmetric key encryption):

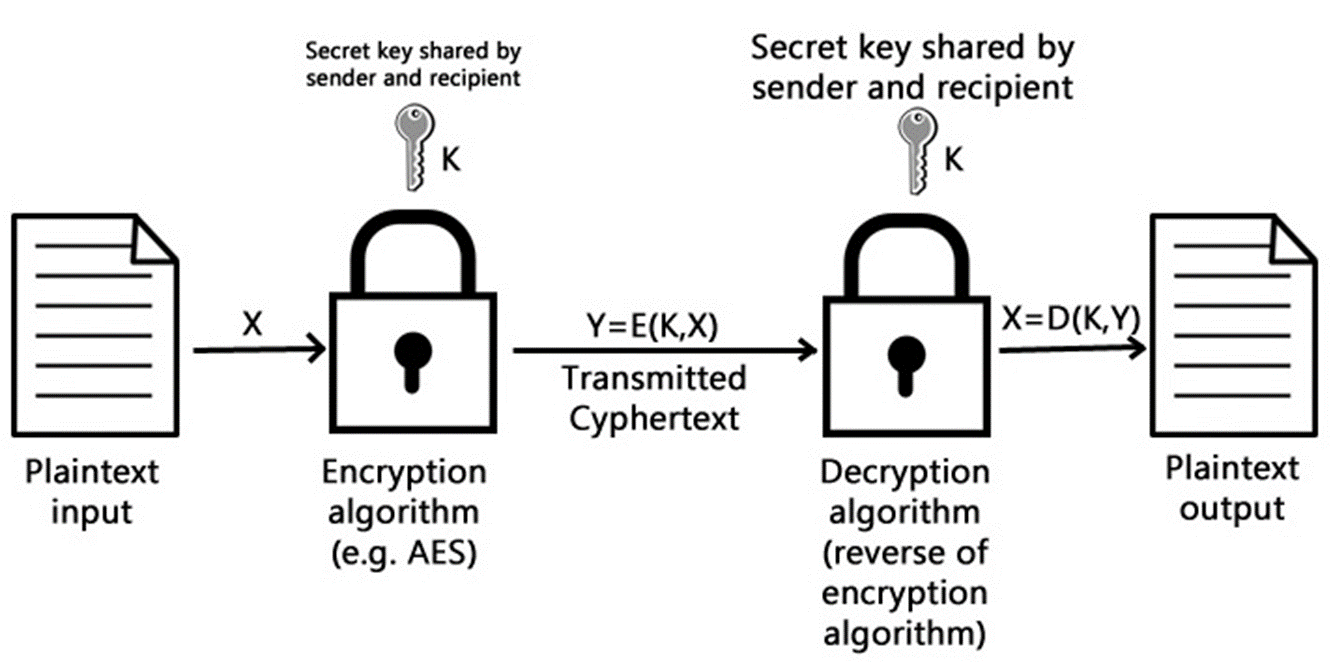
Mã hoá đối xứng sử dụng cùng một khóa để cả mã hóa và giải mã.

**Asymmetric encryption** (asymmetric key encryption):

Mã hoá bất đối xứng sử dụng một khóa dùng để mã hóa (public key), một khóa khác dùng để giải mã (private key).

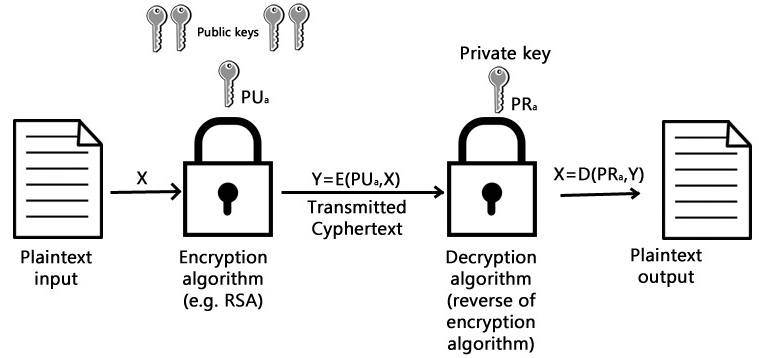
**1.2 Draw an encoding and decoding diagram.**

**Symmetric Encryption**



*Encoding and decoding diagram Symmetric Encryption*

**ASymmetric Encryption**

**

*Encoding and decoding diagram Asymmetric Encryption*

**1.3 Compare.**

|  |  |  |
| --- | --- | --- |
| **Nội dung** | **Mã hoá đối xứng** | **Mã hoá bất đối xứng** |
| **Sử dụng khóa** | Sử dụng cùng một khóa cho cả quá trình mã hóa và giải mã. | Sử dụng một cặp khóa: khóa công khai cho mã hóa và khóa riêng cho giải mã. |
| **Tốc độ** | Nhanh hơn so với mã hóa không đối xứng do đơn giản hóa thuật toán. | Chậm hơn do phức tạp của thuật toán và độ dài của khóa. |
| **An toàn** | An toàn nếu khóa được quản lý cẩn thận. Khi ai đó có khóa, họ có thể mã hóa và giải mã dữ liệu. | An toàn hơn vì khóa công khai có thể được chia sẻ mà không tiết lộ khóa riêng. Nó hữu ích cho việc xác thực và trao đổi khóa bí mật |
| **Sử dụng thường ngày** | Thường được sử dụng cho việc mã hóa lượng lớn dữ liệu, ví dụ như mã hóa ổ cứng hoặc tệp tin. | Thường được sử dụng cho việc trao đổi khóa bí mật, chữ ký số và xác thực trong giao tiếp an toàn. |
| **Bảo mật trong trường hợp mất khóa** | Nếu khóa bị mất hoặc bị lộ, toàn bộ hệ thống mã hóa có thể bị đe dọa | Mất khóa công khai không ảnh hưởng đến tính bí mật của dữ liệu, chỉ ảnh hưởng đến việc trao đổi khóa mới |
| **Điểm yếu** | Đòi hỏi quản lý khóa hiệu quả và an toàn | Chậm hơn và đòi hỏi quản lý khóa công khai một cách cẩn thận. |

**1.4 Write a program to encrypt and decrypt using asymmetric by C# language**

**Symmetric Encryption – tên ứng dụng: AESExample**

**ASymmetric Encryption – tên ứng dụng: RSAExample**

**Question 2 (2 points). Security Vulnerability**

**1. Concept.**

Một lỗ hổng bảo mật là một sai sót hoặc điểm yếu trong hệ thống hoặc mạng cho phép một cuộc tấn công gây hại đến hệ thống hoặc mạng theo một cách nào đó, như:

Cho phép người dùng không được ủy quyền truy cập vào hệ thống hoặc mạng.

Gây suy giảm hiệu suất của hệ thống hoặc mạng.

Gây hại hoặc thay đổi dữ liệu được lưu trữ bởi hệ thống hoặc mạng.

**2. SQL Injection and how to prevent it.**

SQL injection thực hiện bằng cách thêm các câu lệnh SQL để làm sai lệch đi các câu truy vấn ban đầu, dùng SQL injection có thể lấy dữ liệu trong database, thậm chí xoá dữ liệu, xoá database…

**Ngăn chặn**

1. Sử dụng parameter thay vì cộng chuỗi. Nếu dữ liệu truyền vào không hợp pháp, SQL Engine sẽ tự động báo lỗi, ta không cần dùng code để check.

var stmt = "SELECT \* FROM Users WHERE Username = ? AND Password = ?";

stmt->bind\_param("ss", username, password);

2. Loại bỏ hoàn toàn hoặc thay thế các ký tự khả nghi như ở trên có thể thấy ký tự khả nghi là / và ;. Có thể tự viết hàm để thay thế hoặc dùng  mysqli\_real\_escape\_string có sẵn.

3. Sử dụng thư viện/function được cung cấp bởi framework.

4. Không hiển thị exception, message lỗi: Khi có lỗi, ta chỉ hiện thông báo lỗi chứ đừng hiển thị đầy đủ thông tin về lỗi, tránh để lợi dụng, tìm ra cấu trúc database.

5. Phân quyền rõ ràng trong DB: Nếu chỉ truy cập dữ liệu từ một số bảng, hãy tạo một account trong DB, gán quyền truy cập cho account đó chứ đừng dùng account root hay sa. Lúc này, dù có inject được sql cũng không thể đọc dữ liệu từ các bảng chính, sửa hay xoá dữ liệu.

6. Backup dữ liệu thường xuyên: Dữ liệu phải thường xuyên được backup để nếu có bị xoá thì ta vẫn có thể khôi phục được.

**Question 3. Authentication**

E dùng Hash mã hoá chuỗi. Tên Ứng dụng là **HashExample**.