**Câu 1:**

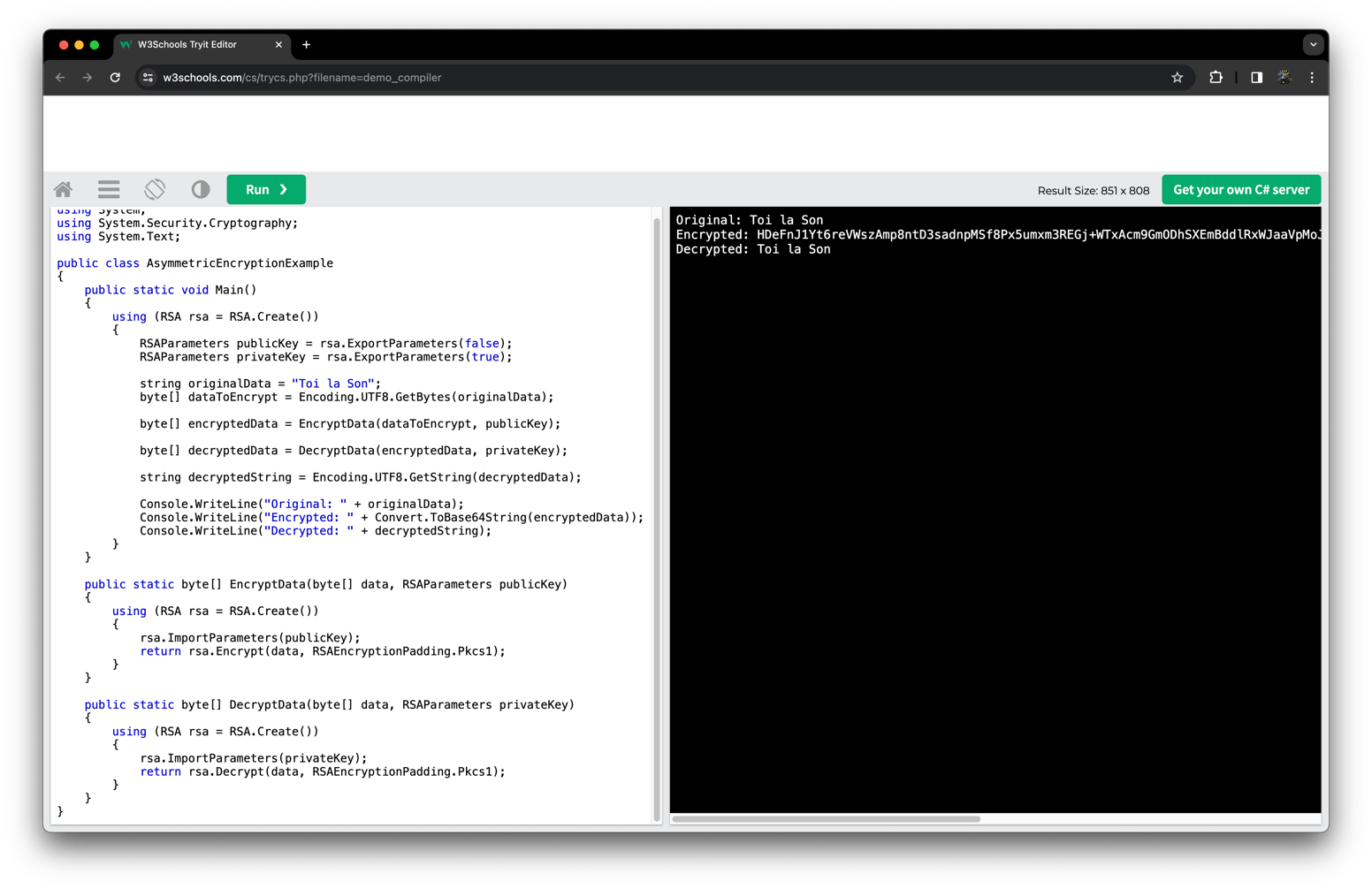
Mã hoá đối xứng và bất đối xứng

* 1. **Khái niệm**
  2. **Sơ đồ mã hoá và giải mã**
  3. **So sánh**

Mã hoá đối xứng và mã hoá bất đối xứng là hai phương pháp mã hóa thông tin được sử dụng trong bảo mật thông tin. Khái niệm, sơ đồ mã hoá, giải mã và so sánh mã hoá đối xứng và mã hoá bất đối xứng được thể hiện trong bảng sau:

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Mục** | **Mã hoá đối xứng** | **Mã hoá bất đối xứng** |
| 1.1 | Khái niệm | Mã hoá đối xứng là phương pháp mã hoá thông tin, trong đó cùng một khoá được sử dụng để mã hoá và giải mã dữ liệu. Khi dữ liệu được mã hoá, nó được chuyển đổi từ dạng thông thường thành dạng không đọc được (ciphertext) bằng cách ánh xạ từng phần của thông tin theo một thuật toán mã hoá. Khi nhận được, khoá được sử dụng để giải mã ciphertext và chuyển đổi nó trở lại thành dữ liệu ban đầu.  Ví dụ phổ biến cho mã hoá đối xứng là AES (Advanced Encryption Standard) và DES (Data Encryption Standard) | Mã hoá bất đối xứng là phương pháp mã hoá thông tin, trong đó sử dụng cặp khoá: khoá công khai (public key) và khoá bí mật (private key). Khoá công khai được chia sẻ rộng rãi và được sử dụng để mã hoá dữ liệu, trong khi khoá bí mật được giữ bởi người nhận và được sử dụng để giải mã thông tin đã được mã hoá. Thuật toán mã hoá bất đối xứng phổ biến nhất là RSA, trong đó khoá công khai có thể được chia sẻ công khai trong khi chỉ người nhận có thể giải mã với khoá bí mật của họ |
| 1.2 | Sơ đồ mã hoá và giải mã | A screenshot of a phone  Description automatically generated | A screenshot of a computer  Description automatically generated |
| 1.3 | So sánh | **Mã hoá đối xứng** | **Mã hoá bất đối xứng** |
| 1.3.1 | Thành phần | •Bản rõ (plaintext-M): bản tin được sinh ra bởi bên gửi  •Bản mật (ciphertext-C): bản tin che giấu thông tin của bản rõ, được gửi tới bên nhận qua một kênh không bí mật  •Khóa (Ks): nó là giá trị ngẫu nhiên và bí mật được chia sẻ giữa các bên trao đổi thông tin và được tạo ra từ:  Bên thứ 3 được tin cậy tạo và phân phối tới bên gửi và bên nhận  Hoặc, bên gửi tạo ra và chuyển cho bên nhận  •Mã hóa (encrypt-E): C = E(KS, M)  •Giải mã (decrypt): M = D(KS, C) = D(KS, E(KS, M)) | •Bản rõ (plaintext-M): bản tin được sinh ra bởi bên gửi  •Bản mật (ciphertext-C): bản tin che giấu thông tin của bản rõ, được gửi tới bên nhận qua một kênh không bí mật  •Khóa: Bên nhận có 1 cặp khóa:  Khóa công khai (Kub) : công bố cho tất cả mọi người biết (kể cả hacker)  Khóa riêng (Krb) : bên nhận giữ bí mật, không chia sẻ cho bất kỳ ai  •Mã hóa (encrypt-E): C = E(Kub, M)  •Giải mã (decrypt): M = D(Krb, C) = D(Krb, E(Kub, M))  Yêu cầu đối với cặp khóa (Kub, Krb) là:  •Hoàn toàn ngẫu nhiên  •Có quan hệ về mặt toán học 1-1.  •Nếu chỉ có giá trị của Kub không thể tính được Krb.  •Krb phải được giữ mật hoàn toàn. |
| 1.3.2 | Cơ chế hoạt động | •Người gửi sử dụng khóa chung (Ks) để mã hóa thông tin rồi gửi cho nguời nhận.  •Người nhận nhận được thông tin đó sẽ dùng chính khóa chung (Ks) để giải mã. | •Người gửi(A) gửi thông tin đã được mã hóa bằng khóa công khai (Kub) của người nhận(B) thông qua kênh truyền tin không bí mật  •Người nhận(B) nhận được thông tin đó sẽ giải mã bằng khóa riêng (Krb) của mình.  •Hacker cũng sẽ biết khóa công khai (Kub) của B tuy nhiên do không có khóa riêng (Krb) nên Hacker không thể xem được thông tin gửi |
| 1.3.3 | Nhược điểm | Do dùng chung khóa để mã hóa và giải mã => nếu bị mất hoặc bị đánh cắp bởi hacker sẽ bị lộ thông tin, bảo mật không cao.  Cần kênh mật để chia sẻ khóa bí mật giữa các bên  Khó ứng dụng trong các hệ thống mở.  Không thể dùng cho mục đích xác thực hay mục đích chống thoái thác được. | Chưa có kênh an toàn để chia sẻ khóa => Khả năng bị tấn công dạng tấn công người đứng giữa (man in the middle attack ). |
| 1.3.4 | Ưu điểm | Đơn giản, dễ làm | Không cần chia sẻ khóa mã hóa(khóa công khai) một cách bí mật => Dễ dàng ứng dụng trong các hệ thống mở.  Khóa giải mã(khóa riêng) chỉ có B biết => An toàn hơn, có thể xác thực nguồn gốc thông tin  n phần tử chỉ cần n cặp khóa |

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



**Câu 2:**

2.1. Khái niệm lỗ hổng bảo mật:

Lỗ hổng bảo mật là những điểm yếu nằm trong thiết kế và cấu hình của hệ thống, lỗi của lập trình viên hoặc sơ suất trong quá trình vận hành.

*Cách thức hoạt động*

•Hacker sử dụng các công cụ dò quét để phát hiện một loạt các website có cấu hình bảo mật kém hoặc website trên các nền tảng phổ biến như WordPress hay Joomla có các lỗ hổng đã được công bố nhưng chưa được chủ website xử lý.

•Tin tặc sẽ lợi dụng chúng để tấn công, cài đặt mã độc và phá hoại các website.

2.2. SQL Injection và cách phòng tránh

SQL Injection là một kỹ thuật tấn công mạng phổ biến mà kẻ tấn công sử dụng để chèn các đoạn mã SQL độc hại vào các truy vấn SQL được tạo bởi ứng dụng web. Điều này có thể dẫn đến việc xâm nhập vào cơ sở dữ liệu, thậm chí là kiểm soát hoàn toàn cơ sở dữ liệu. Đây là một trong những lỗ hổng bảo mật nguy hiểm và phổ biến trong các ứng dụng web.

Một số cách phòng tránh SQL Injection:

2.2.1. Sử dụng Prepared Statements và Parametrized Queries: Thay vì tạo truy vấn SQL bằng cách ghép chuỗi, sử dụng prepared statements hoặc parametrized queries. Điều này sẽ tách dữ liệu người dùng và câu lệnh SQL, ngăn chặn việc chèn mã SQL độc hại.

2.2.2. Kiểm tra và Xử lý Dữ liệu Người Dùng: Luôn kiểm tra và xử lý dữ liệu được nhập từ người dùng trước khi sử dụng trong truy vấn SQL. Sử dụng các hàm kiểm tra dữ liệu (ví dụ như escape strings) để đảm bảo rằng dữ liệu không chứa các ký tự đặc biệt có thể được sử dụng để phá vỡ câu lệnh SQL.

2.2.3. Nguyên Tắc Nguyên Mẫu Lối Chạy ít Quyền Nhất (Principle of Least Privilege): Đảm bảo rằng người dùng của ứng dụng chỉ có quyền truy cập cần thiết vào cơ sở dữ liệu. Hạn chế quyền truy cập và thực hiện các biện pháp an ninh để giảm thiểu tổn thất khi có cuộc tấn công.

2.2.4. Sử Dụng Công Cụ Bảo Mật: Sử dụng các công cụ bảo mật và firewall để giám sát và ngăn chặn các loại tấn công SQL Injection.

2.2.5. Cập Nhật Hệ Thống và Thư Viện: Luôn cập nhật các hệ thống và thư viện của bạn để bảo vệ chúng khỏi các lỗ hổng bảo mật đã biết.

2.2.6. Chỉ Hiển Thị Thông Tin Cần Thiết: Đừng hiển thị thông tin cụ thể về lỗi SQL cho người dùng cuối. Thông tin này có thể được kẻ tấn công sử dụng để tìm ra lỗ hổng và thực hiện cuộc tấn công tiếp theo.

**Câu 3: Authentication**

