**1. GenerateKeys(RSAForm form)**

Chức năng: Sinh các khóa công khai và bí mật cho RSA.

* p, q: hai số nguyên tố ngẫu nhiên.
* N = p \* q: modulus chung.
* phi = (p-1)(q-1): hàm Euler, cần thiết để tính nghịch đảo.
* e: số nguyên lẻ bắt đầu từ 3, tăng dần đến khi GCD(e, phi) == 1.
* d: nghịch đảo modular của e mod phi → dùng để giải mã.

**2. Encrypt(RSAForm form)**

Chức năng: Mã hóa bản rõ (số hoặc chuỗi).

* Nếu txtPlainInput là số: dùng BigInteger.ModPow(message, e, N).
* Nếu là chuỗi: gọi EncryptText(form) để mã hóa từng ký tự.

**3.Decrypt(RSAForm form)**

Chức năng: Giải mã bản mã (số hoặc chuỗi).

* Nếu txtCipher là số → ModPow(cipher, d, N) để giải mã.
* Nếu là chuỗi mã hóa từng ký tự → gọi DecryptText(form).

**4. EncryptText()**

Chức năng: Mã hóa từng ký tự chuỗi.

* Duyệt qua từng char, chuyển thành số (BigInteger).
* Áp dụng công thức mã hóa RSA: c = m^e mod N.
* Lưu từng số mã hóa vào danh sách, ghép chuỗi bằng dấu cách " ".

**5. DecryptText()**

Chức năng: Giải mã bản mã chuỗi về chuỗi rõ.

* Chia txtCipher theo " ", lấy từng mã số.
* Tính m = c^d mod N để lấy mã Unicode.
* Chuyển về ký tự và ghép lại thành chuỗi.

**6. GeneratePrime() + IsPrime()**

Chức năng: Sinh ngẫu nhiên số nguyên tố nhỏ (dùng cho demo).

* Tạo số ngẫu nhiên 3 byte → ép về số nguyên dương.
* Kiểm tra nguyên tố bằng chia thử.
* Lưu ý: Với mục đích học tập, sinh số nguyên tố nhỏ là ok. Với thực tế cần thuật toán mạnh hơn như Miller-Rabin.

**7. ModInverse()**

Chức năng: Tính nghịch đảo modular d = e⁻¹ mod phi.

* Dùng thuật toán Euclid mở rộng.
* Cần đảm bảo m != 0 để tránh lỗi DivideByZeroException.

**3. Hàm tính Ước Chung Lớn Nhất (GCD)**

private static BigInteger GCD(BigInteger a, BigInteger b)

* Tính GCD của hai số a và b bằng thuật toán Euclid.
* Lặp đến khi b = 0.

**4. Hàm tạo khóa**

**public static void GenerateKeys(RSAForm form)**

* Tạo hai số nguyên tố p, q.
* Tính N = p \* q.
* Tính phi = (p - 1) \* (q - 1) (phi Euler).
* Chọn e ban đầu = 3, sau đó tăng dần cho tới khi GCD(e, phi) = 1.
* Tính d = ModInverse(e, phi) là khóa bí mật.
* Gán các giá trị lên giao diện form.

**5. Mã hóa (Encrypt)**

**public static void Encrypt(RSAForm form)**

* Nếu input là số thì dùng BigInteger.ModPow(m, e, N) để mã hóa.
* Nếu input là chuỗi (không parse được sang số), gọi EncryptText.

**6. Giải mã (Decrypt)**

**public static void Decrypt(RSAForm form)**

* Nếu input là số thì dùng BigInteger.ModPow(c, d, N) để giải mã.
* Nếu input là chuỗi mã hóa, gọi DecryptText.

**7. Mã hóa chuỗi**

**public static void EncryptText(RSAForm form)**

* Duyệt từng ký tự trong chuỗi.
* Convert char → int → BigInteger.
* Mã hóa từng ký tự bằng BigInteger.ModPow**.**
* Nối lại thành một chuỗi (cách nhau bởi dấu cách).

**8. Giải mã chuỗi**

**public static void DecryptText(RSAForm form)**

* Tách chuỗi mã hóa thành các phần tử bằng Split(' ').
* Duyệt từng phần tử:
* Giải mã mỗi phần tử BigInteger.ModPow(c, d, N).
* Convert int → char, thêm vào StringBuilder.

**9. Tạo số nguyên tố ngẫu nhiên**

**private static BigInteger GeneratePrime()**

* Tạo 3 byte ngẫu nhiên → chuyển thành số.
* Lặp đến khi số đó là số nguyên tố.

**10. Kiểm tra số nguyên tố**

**private static bool IsPrime(BigInteger number)**

* Trả về false nếu nhỏ hơn 2 hoặc chia hết cho 2.
* Duyệt từ 3 đến √n, nếu chia hết thì không phải nguyên tố.

**11. Tính số nghịch đảo modulo**

**private static BigInteger ModInverse(BigInteger a, BigInteger m)**

* Dùng thuật toán Euclid mở rộng để tìm d sao cho (e \* d) mod phi = 1.
* Kiểm tra chia 0 để tránh lỗi.
* Nếu x1 < 0, cộng thêm m0 để kết quả dương.

Ký số

**🔒 Tính năng nâng cao đã thêm:**

* ✅ Băm SHA-256 cho thông điệp thay vì chuyển chuỗi trực tiếp sang số.
* ✅ Đảm bảo BigInteger luôn là **số dương** (bằng cách thêm 1 byte 0 vào cuối mảng).
* ✅ Xử lý tình huống n không trùng giữa hai giai đoạn.
* RSA thường không hoạt động trực tiếp trên bản băm lớn hơn n, vì n trong thực tế rất lớn (1024+ bit). Nhưng với demo nhỏ như n = 3233, bạn phải lấy hash % n để đảm bảo 0 <= hash < n.