**Implementation of RSA Algorithm**

**Network Security Assignment - CS1072**

**Submitted By**

***Mohamed Shalman Kursheeth K (CS22B1032)***

**TO**

**Dr. Narendran Rajagopalan**

***Associate Professor***

***Department of Computer Science and Engineering***

***National Institute of Technology Puducherry***

***Karaikal – 609609***

****

**DEPARTMENT OF**

**COMPUTER SCIENCE AND ENGINEERING**

**NATIONAL INSTITUTE OF TECHNOLOGY PUDUCHERRY**

**KARAIKAL – 609 609**

**APRIL 2025**

Public-key cryptography, or asymmetric cryptography, is a cryptographic system that uses pairs of keys: public keys, which may be disseminated widely, and private keys,which are known only to the owner. The generation of such keys depends on cryptographic algorithms based on mathematical problems to produce one-way functions. Effective security only requires keeping the private key private; the public key can be openly distributed without compromising security.

In such a system, any person can encrypt a message using the receiver's public key, but that encrypted message can only be decrypted with the receiver's private key.

**RSA Algorithm**

RSA (Rivest–Shamir–Adleman) is one of the first public-key cryptosystems and is widely used for secure data transmission. In such a cryptosystem, the encryption key is public and distinct from the decryption key which is kept secret (private). In RSA, this asymmetry is based on the practical difficulty of factoring the product of two large prime numbers, the "factoring problem". The acronym RSA is the initial letters of the surnames of Ron Rivest, Adi Shamir, and Leonard Adleman, who publicly described the algorithm in 1977. Clifford Cocks, an English mathematician working for the British intelligence agency Government Communications Headquarters (GCHQ), had developed an equivalent system in 1973, which was not declassified until 1997.

A user of RSA creates and then publishes a public key based on two large prime numbers, along with an auxiliary value. The prime numbers must be kept secret. Anyone can use the public key to encrypt a message, but only someone with knowledge of the prime numbers can decode the message. Breaking RSA encryption is known as the RSA problem. Whether it is as difficult as the factoring problem is an open question. There are no published methods to defeat the system if a large enough key is used.

RSA is a relatively slow algorithm, and because of this, it is less commonly used to directly encrypt user data. More often, RSA passes encrypted shared keys for symmetric key cryptography which in turn can perform bulk encryption-decryption operations at much higher speed.

**RSA Algorithm**

* Pick two large primes p ,q.
* Compute n= pq and φ(n)=lcm(p−1,q−1)
* Choose a public key e such that 1<e<φ(n) and gcd (e, φ(n)) =1
* Calculate d such that de≡1(mod φ(n))
* Let the message **key** be m
* Encrypt:  c ≡ m\*\*e (mod n)
* Decrypt:  m ≡ c\*\*d (mod n)

A white background with black text

AI-generated content may be incorrect.

**Imports and Helper Functions**

The program first imports necessary libraries:

* math.sqrt is used to compute square roots.
* random is used to generate random numbers.
* randint is imported with an alias rand, mimicking C++'s rand() function.

A screenshot of a computer program

AI-generated content may be incorrect.

Several helper functions are defined:

1. **gcd(a, b)**: Computes the greatest common divisor (GCD) using recursion.
2. **mod\_inverse(a, m)**: Finds the modular inverse of a modulo m. This is used to compute the private key.
3. **isprime(n)**: Checks whether a given number n is prime. It does this by checking divisibility up to sqrt(n), skipping even numbers

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

**Generating Prime Numbers**

Initially, two random numbers p and q are selected. These numbers should ideally be prime, but the program does not check this explicitly before use.

The function **generate\_keypair(p, q, keysize)** is responsible for generating RSA key pairs:

* It first defines the valid range for n = p \* q based on the given key size.
* A list of prime numbers is generated in the specified range.
* It selects two prime numbers, p and q, such that p \* q falls within the valid range.

**Computing RSA Key Components**

After selecting p and q, the function computes:

1. **n = p \* q**: The modulus used in encryption and decryption.
2. **phi = (p - 1) \* (q - 1)**: Euler’s totient function, which is crucial for key generation.
3. **Selecting e (Public Exponent)**:
   * e is chosen randomly such that 1 < e < phi and gcd(e, phi) = 1.
4. **Computing d (Private Exponent)**:
   * d is the modular inverse of e modulo phi, ensuring e \* d ≡ 1 (mod phi).

The function returns a public key (e, n) and a private key (d, n).

A screenshot of a computer program

AI-generated content may be incorrect.

**Encryption and Decryption**

Two functions handle message encryption and decryption:

1. **encrypt(msg\_plaintext, package)**:
   * Uses the public key (e, n) to encrypt each character in the message using pow(ord(c), e, n).
   * ord(c) converts a character into its ASCII equivalent before exponentiation.
   * The result is a list of integers representing the encrypted message.
2. **decrypt(msg\_ciphertext, package)**:
   * Uses the private key (d, n) to decrypt the message using pow(c, d, n).
   * The result is converted back to characters and joined into a string.

A computer code with text

AI-generated content may be incorrect.

**Driver Code**

The program starts execution in the if \_\_name\_\_ == "\_\_main\_\_" block:

1. It asks for a bit length from the user.
2. Calls generate\_keypair to create the public and private keys.
3. Prints the keys and asks the user for a message.
4. Encrypts the message and displays the encrypted form.
5. Decrypts the encrypted message and prints the original text.

OUTPUT:

A computer screen shot of a computer code

AI-generated content may be incorrect.