|  |  |  |  |
| --- | --- | --- | --- |
| **Course Name:** | **Applied Cryptography 116U01E628** | **Semester:** | **VI** |
| **Date of Performance:** | **26/03/2025** | **DIV/ Batch No:** | **C - 3** |
| **Student Name:** | **Romil Lodaya** | **Roll No:** | **16010122096** |

**Experiment No:7**

**Title: Implementation of RSA algorithm**

|  |
| --- |
| **Aim and Objective of the Experiment:** |
| Implementation of RSA algorithm |

|  |
| --- |
| **COs to be achieved:** |
| **CO2: Demonstrate and implement various Cryptographic Algorithms for securing systems.** |

|  |
| --- |
| **Books/ Journals/ Websites referred:** |
| 1. Stallings, W., Cryptography and Network Security: Principles and Practice, Second edition, Person Education 2. Forouzan, B. A. (2018). Cryptography and Network Security. McGraw-Hill Education. |

|  |
| --- |
| **Theory:** Explain the following. |
| Explain the requirement of asymmetric key cryptography:  **Requirement of Asymmetric Key Cryptography (Public Key Cryptography)**  Asymmetric key cryptography, also known as **public-key cryptography**, uses two keys:   * **Public Key** – Available to everyone. * **Private Key** – Kept secret by the owner.   **1. Secure Key Exchange**   * In symmetric key cryptography, both parties need to share the same secret key securely. * Asymmetric cryptography solves this by enabling **secure key exchange** without transmitting the private key. * Example: **Diffie-Hellman key exchange**.   **2. Confidentiality and Privacy**   * Ensures that only the intended recipient can decrypt the message. * **Process:**   + Sender encrypts the message using the recipient’s **public key**.   + Only the recipient can decrypt it using their **private key**. * Even if someone intercepts the message, they cannot decrypt it without the private key.   **3. Digital Signatures (Authentication and Integrity)**   * Provides **authentication** and **integrity** to ensure that the message was not altered. * **Process:**   + Sender encrypts the hash of the message with their **private key** to generate a **digital signature**.   + The recipient verifies the signature using the sender's **public key**. * Protects against forgery and tampering.   **4. Non-repudiation**   * Ensures that the sender cannot deny sending a message. * Since only the sender’s **private key** can create a valid digital signature, they cannot deny ownership of the message.   **5. Scalability for Large Networks**   * Asymmetric encryption scales better for large environments. * In symmetric systems, the number of keys grows exponentially:   Number of keys=n(n−1)2\text{Number of keys} = \frac{n(n-1)}{2}Number of keys=2n(n−1)​   * In asymmetric cryptography, only **two keys per user** are required.   **6. Secure Web Communication (HTTPS/SSL/TLS)**   * Asymmetric cryptography is widely used in securing web communications. * **SSL/TLS protocols** use asymmetric encryption to establish secure sessions between clients and servers.   **7. Protection Against Key Compromise**   * In case of a compromised private key, only the messages encrypted with that key are at risk. * With symmetric encryption, if a key is compromised, all communication using that key is vulnerable.   RSA(Rivest-Shamir-Adleman) Algorithm is an asymmetric or public-key cryptography algorithm which means it works on two different keys: Public Key and Private Key. The Public Key is used for encryption and is known to everyone, while the Private Key is used for decryption and must be kept secret by the receiver. RSA Algorithm is named after Ron Rivest, Adi Shamir and Leonard Adleman, who published the algorithm in 1977.  RSA Algorithm is based on factorization of large number and modular arithmetic for encrypting and decrypting data. It consists of three main stages:   1. Key Generation: Creating Public and Private Keys 2. Encryption: Sender encrypts the data using Public Key to get cipher text. 3. Decryption: Decrypting the cipher text using Private Key to get the original data.   1. Key Generation   * Choose two large prime numbers, say p and q. These prime numbers should be kept secret. * Calculate the product of primes, n = p \* q. This product is part of the public as well as the private key. * Calculate Euler’s Totien Function Φ(n) as Φ(n) = Φ(p \* q) = Φ(p) \* Φ(q) = (p – 1) \* (q – 1). * Choose encryption exponent e, such that   + 1 < e < Φ(n), and   + gcd(e, Φ(n)) = 1, that is e should be co-prime with Φ(n). * Calculate decryption exponent d, such that   + (d \* e) ≡ 1 mod Φ(n), d is modular multiplicative inverse of e mod Φ(n). Some common methods to calculate multiplicative inverse are the extended Euclidean Algorithm, Fermat’s Little Theorem, etc.   + We can have multiple values of d satisfying (d \* e) ≡ 1 mod Φ(n) but it does not matter which value we choose as all of them are valid keys and will result in same message on decryption.   + Finally, the Public Key = (n, e) and the Private Key = (n, d).   2. Encryption: To encrypt a message M, it is first converted to numerical representation using ASCII and other encoding schemes. Now, use the public key (n, e) to encrypt the message and get the cipher text using the formula:  C = Me mod n, where C is the Cipher text and e and n are parts of public key.  3. Decryption  To decrypt the cipher text C, use the private key (n, d) and get the original data using the formula:  M = Cd  mod n, where M is the message and d and n are parts of private key. |

|  |
| --- |
| **Code and Output :** |
| **CODE:**  #include <bits/stdc++.h>  using namespace std;  long long gcd(long long a, long long b)  {     while (b != 0)     {        long long temp = b;        b = a % b;        a = temp;     }     return a;  }  long long modInverse(long long a, long long m)  {     a = a % m;     for (long long x = 1; x < m; x++)     {        if ((a \* x) % m == 1)        {           return x;        }     }     return -1;  }  long long power(long long base, long long exp, long long mod)  {     long long result = 1;     base = base % mod;     while (exp > 0)     {        if (exp % 2 == 1)           result = (result \* base) % mod;        exp = exp >> 1;        base = (base \* base) % mod;     }     return result;  }  int main()  {     long long p, q, n, phi, e, d, msg, encrypted, decrypted;     cout << "Enter prime number p: ";     cin >> p;     cout << "Enter prime number q: ";     cin >> q;     n = p \* q;     phi = (p - 1) \* (q - 1);     cout << "Enter message to encrypt (integer): ";     cin >> msg;     e = 2;     while (e < phi && gcd(e, phi) != 1)     {        e++;     }     d = modInverse(e, phi);     encrypted = power(msg, e, n);     decrypted = power(encrypted, d, n);     cout << "Original Message: " << msg << endl;     cout << "Encrypted Message: " << encrypted << endl;     cout << "Decrypted Message: " << decrypted << endl;     return 0;  }  **OUTPUT:** |

|  |
| --- |
| **Post Lab Subjective/Objective type Questions:** |
| 1. Solve one example using RSA algorithm.      1. Write Applications of RSA.   **1. Secure Data Transmission**   * RSA encrypts sensitive information transmitted over insecure networks. * Used in **email encryption, VPNs, and instant messaging**.   **2. Digital Signatures**   * Ensures **authentication, integrity, and non-repudiation**. * Used in:   + **E-commerce platforms** for secure transactions.   + **Software distribution** to verify authenticity.   + **Blockchain technology** for validating transactions.   **3. HTTPS/SSL/TLS Protocols**   * RSA is used during the **handshake phase** of TLS/SSL protocols to:   + Establish a secure session between the browser and the web server.   + Exchange keys securely before switching to symmetric encryption for faster communication.   **4. Secure Email Communication (PGP/GPG)**   * RSA encrypts emails and digital signatures in protocols such as:   + **Pretty Good Privacy (PGP)**   + **GNU Privacy Guard (GPG)**   **5. Digital Certificates and PKI (Public Key Infrastructure)**   * RSA is used to:   + Generate and verify digital certificates.   + Authenticate the identity of websites and users.   + Provide trust in secure communications through certificate authorities (CAs).   **6. Blockchain and Cryptocurrencies**   * RSA is used for key management and securing wallets in blockchain networks. * It ensures the confidentiality and authenticity of transactions.   **7. Secure Software Updates**   * RSA ensures that software updates come from trusted sources, preventing malicious updates.  1. Comment on the strengths and weaknesses of RSA.   **Strengths of RSA**  **1. Strong Security**   * RSA is based on the **difficulty of prime factorization** of large numbers. * As key sizes increase, the algorithm becomes computationally infeasible to break.   **2. Asymmetric Nature**   * No need to share a secret key between communicating parties. * Public keys can be freely distributed while private keys remain secure.   **3. Confidentiality and Integrity**   * Provides **encryption and digital signatures**. * Ensures that data is not only private but also authenticated.   **4. Non-Repudiation**   * Digital signatures prevent the sender from denying sending the message.   **5. Wide Adoption and Proven Security**   * RSA has been rigorously tested over decades, making it a trusted encryption standard.   **Weaknesses of RSA**  **1. Computationally Slow**   * RSA is significantly slower than symmetric encryption algorithms (like AES). * Encryption and decryption of large messages can be time-consuming.   **2. Vulnerable to Brute-Force Attacks (if key size is small)**   * RSA with small key sizes (e.g., 512-bit or 1024-bit) is vulnerable to **brute-force attacks**. * Modern standards recommend at least **2048-bit** keys.   **3. Susceptible to Chosen Plaintext Attacks (CPA)**   * RSA can be vulnerable to attacks if **padding schemes** are not properly implemented (e.g., RSA without padding can be exploited through attacks like **Bleichenbacher’s attack**).   **4. Key Management Complexity**   * Public and private key management is more complex than symmetric systems. * Secure key storage and revocation are challenging.   **5. Not Suitable for Bulk Data Encryption**   * Due to its computational inefficiency, RSA is not ideal for encrypting large amounts of data. * It is typically used to **encrypt symmetric session keys** instead. |

|  |
| --- |
| **Conclusion:** |
| The RSA algorithm ensures secure communication by encrypting data, providing confidentiality, integrity, and authentication through asymmetric encryption. |