|  |  |  |  |
| --- | --- | --- | --- |
| **Course Name:** | **Applied Cryptography 116U01E628** | **Semester:** | **VI** |
| **Date of Performance:** | **05 / 02 / 2025** | **DIV/ Batch No:** | **C - 3** |
| **Student Name:** | **Romil Lodaya** | **Roll No:** | **16010122096** |

**Experiment No: 5**

**Title: Diffie Hellman Key exchange Algorithm**

|  |
| --- |
| **Aim and Objective of the Experiment:** |
| Implementation of Diffie Hellman Key exchange Algorithm   1. The objective of this experiment is to write a program for the Diffie-Hellman Key exchange algorithm and verify if the secret key computed by Alice and Bob are the same or not. 2. Attempt the Man in the middle attack variation |

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

|  |
| --- |
| **Books/ Journals/ Websites referred:** |
| 1. GeeksforGeeks |

|  |
| --- |
| **Theory:** |
| **Abstract**:-  The Diffie–Hellman (DH) Algorithm is a key-exchange protocol that enables two parties communicating over public channel to establish a mutual secret without it being transmitted over the Internet. DH enables the two to use a public key to encrypt and decrypt their conversation or data using symmetric cryptography.  The sender and receiver don't need any prior knowledge of each other. Once the keys are exchanged, the communication of data can be done through an insecure channel. The sharing of the secret key is safe.  **Related Theory: -**  Diffie–Hellman key exchange is a method of securely exchanging cryptographic keys over a public channel and was one of the first public-key protocols as conceived by Ralph Merkle and named after Whitfield Diffie and Martin Hellman. DH is one of the earliest practical examples of public key exchange implemented within the field of cryptography. Published in 1976 by Diffie and Hellman, this is the earliest publicly known work that proposed the idea of a private key and a corresponding public key.    Diffie–Hellman key exchange establishes a shared secret between two parties that can be used for secret communication for exchanging data over a public network. An analogy illustrates the concept of public key exchange by using colours instead of very large numbers.  The process begins by having the two parties, Alice and Bob, publicly agree on an arbitrary starting colour that does not need to be kept secret (but should be different every time). In this example, the colour is yellow. Each person also selects a secret colour that they keep to themselves – in this case, red and blue-green. The crucial part of the process is that Alice and Bob each mix their own secret colour together with their mutually shared colour, resulting in orange-tan and light-blue mixtures respectively, and then publicly exchange the two mixed colours. Finally, each of them mixes the colour they received from the partner with their own private colour. The result is a final colour mixture (yellow-brown in this case) that is identical to the partner's final colour mixture.  If a third party listened to the exchange, it would only know the common colour (yellow) and the first mixed colours (orange-tan and light-blue), but it would be difficult for this party to determine the final secret colour (yellow-brown). Bringing the analogy back to a real-life exchange using large numbers rather than colours, this determination is computationally expensive. It is impossible to compute in a practical amount of time even for modern supercomputers. |

|  |
| --- |
| **Algorithm :** |
| // Step 1: Input Public Parameters  PRINT "Enter the public number P & G:"  READ P, G  // Step 2: Input Private Keys  PRINT "Enter the private keys a (Alice) & b (Bob):"  READ a, b  // Step 3: Calculate Public Keys  publicA = (G^a) MOD P  publicB = (G^b) MOD P  // Step 4: Calculate Shared Secret Key  symKeyA = (publicB^a) MOD P  symKeyB = (publicA^b) MOD P  // Step 5: Output Shared Secret Key  PRINT "Shared secret key:"  PRINT "Alice: ", symKeyA  PRINT "Bob: ", symKeyB  END Diffie-Hellman Key Exchange |

|  |
| --- |
| **Solve a small numerical Example for assigned algorithm(Paste photograph of same) :** |
|  |

|  |
| --- |
| **Code :** |
| #include<bits/stdc++.h>  using namespace std;  int main()  {  cout << "Enter the public number P & G: ";  int p, g;  cin >> p >> g;  cout << "Enter the private keys a(Alice) & b(Bob): ";  int a, b;  cin >> a >> b;  int publicA, publicB;  publicA = (int)pow(g,a) % p;  publicB = (int)pow(g,b) % p;  int symKeyA = (int)pow(publicB, a) % p;  int symKeyB = (int)pow(publicA, b) % p;  cout << endl << "Shared secret key: " << endl;  cout << "Alice: " << symKeyA << endl;  cout << "Bob: " << symKeyB << endl;  return 0;  } |

|  |
| --- |
| **Output:** |
|  |

|  |
| --- |
| **Post Lab Subjective/Objective type Questions:** |
| 1. Comment on weakness(s) of Diffie-Hellman scheme    **Man-in-the-Middle Attacks**:   * The Diffie-Hellman scheme lacks built-in authentication. An attacker can intercept the public keys exchanged between two parties and substitute their own keys, allowing the attacker to establish separate keys with both parties. This enables the attacker to decrypt and alter the communication without either party being aware.    **Static Key Vulnerability**:   * If the same private keys are used across multiple sessions, an attacker who compromises those private keys can derive the shared secret for past communications. This reuse of keys can lead to the exposure of sensitive data even after the initial communication.    **No Authentication**:   * Without any form of authentication, parties cannot be sure of the identities of the other parties involved. This can lead to impersonation and unauthorized access.    **Computational Complexity**:   * The basic Diffie-Hellman exchange can be computationally expensive, especially when using large prime numbers, which can result in performance issues in resource-constrained environments.  1. Suggest at least two methods to eliminate the weakness(s) of D-H scheme.    **Implement Authentication**:   * Use digital signatures or public key infrastructure (PKI) to authenticate the parties involved. Before exchanging public keys, Alice and Bob can sign their keys with their private keys. This way, the recipient can verify that the key indeed belongs to the claimed sender, preventing man-in-the-middle attacks.    **Use Ephemeral Keys**:   * Implement ephemeral (temporary) keys for each session instead of static private keys. Alice and Bob can generate a new private key for each session, ensuring that even if one key is compromised, it only affects that session. This approach limits the exposure of shared secrets to only one communication session.    **Hybrid Cryptography**:   * Combine the Diffie-Hellman key exchange with symmetric encryption and PKI. Once the shared secret is established, it can be used to derive a symmetric key for encrypting the actual messages. Using PKI ensures that public keys are authenticated, reducing the risk of man-in-the-middle attacks.    **Use of Strong Parameters**:   * Ensure that large and secure prime numbers and generators are used in the Diffie-Hellman exchange. Using recommended key sizes (2048 bits or more) can help protect against certain types of attacks, such as those based on computing discrete logarithms. |

|  |
| --- |
| **Conclusion:** |
| The experiment successfully implemented the Diffie-Hellman algorithm, demonstrating key exchange integrity and exposing vulnerabilities through a man-in-the-middle attack. |