**EXCEL ENGINEERING COLLEGE**

**(Autonomous)**

**Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai**

**Accredited by NBA, NAAC with “A+” and Recognised by UGC (2f &12B)**

**KOMARAPALAYAM - 637303**



**DEPARTMENTOF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**

**20AI703-CRYPTOGRAPHY AND NETWORK SECURITY LABORATORY**

**VII SEMESTER- R2020**

**REFERENCE MANUAL**

PREPARED BY

**P.Jayaprabha,**

**AP&Head, AI&DS**

**EXCELENGINEERINGCOLLEGE**

**VISION**

To create competitive human resources in the fields of engineering for the benefit of

society to meet global challenges.

**MISSION**

* To provide a conducive ambience for better learning and to bring creativity in the students.
* To develop sustainable environment for innovative learning to serve the needy.
* To meet global demands for excellence in technical education.
* To train young minds with values, culture, integrity, innovation and leadership.

**DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**

**VISION**

To create better quality technical engineers in computer science and engineering with ethically strong values which cater local and global needs of the society.

**MISSION**

* + To instill quality in engineering education that demands excellence.
  + To initiate desires among the students to work in close cooperation and collaboration with industry and professional bodies.
  + To train the students for developing software and novel software systems.
  + To create ambience for taking initiatives towards entrepreneurship and lifelong learning.

# PROGRAMME EDUCATIONAL OBJECTIVES(PEOs)

1. To provide fundamental knowledge to formulate, solve and analyze engineering problems and pursue higher studies.
2. To develop the ability of the students in comprehending, analyzing and synthesizing data in order to design software and create novel software systems.
3. To inculcate effective communication skills, team skills, professional and ethical attitude in the students for enabling them to relate engineering issues with social issues in a broader context.
4. To provide students managerial and leadership skills so as to make them successfully employed and to demonstrate a pursuit of lifelong learning in multidisciplinary environment.

# PROGRAMME OUTCOMES (POs)

* 1. **Engineering Knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.
  2. **Problem Analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences.
  3. **Design / Development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
  4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods, including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
  5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources and modern engineering and IT tools including prediction and modeling of complex engineering activities with an understanding of the limitations.
  6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
  7. **Environment and Sustainability**: Understand the impact of the professional engineering solutions to societal and environmental contexts and demonstrate the knowledge of, and need for sustainable development.
  8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
  9. **Individual and team work**: Function effectively as an individual and as a member or leader in diverse teams, and in multidisciplinary settings.
  10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations and give and receive clear instructions.
  11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
  12. **Lifelong learning**: Recognize the need for and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change.

Excel Logo1 **Engineering College**

Department of Artificial Intelligence and Data Science

Sub.Code/Name: 20AI703 CRPTOGRAPHY AND NETWORK SECURITY LAB Sem./Year : VII / IV

AcademicYear:2023- 24 Batch:2020–2024

**Syllabus**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **List of Exercises** | **CO**  **Mapping** | **RBT** |
| **1** | Write a program to perform encryption and decryption using the following algorithms  a) Substitution Ciphers b) Transposition Ciphers | CO1 | Apply |
| **2** | Write a program to implement the DES algorithm logic. | CO2 | Analyze |
| **3** | Write a program to implement RSA algorithm. | CO3 | Analyze |
| **4** | Implement the Diffie - Hellman Key Exchange algorithm for a given problem | CO3 | Apply |
| **5** | Calculate the message digest of a text using the SHA-1 algorithm | CO4 | Apply |
| **6** | Demonstrate intrusion detection system (ids) using any tool  eg. Snort or any other s/w | CO5 | Analyze |

# LIST OF EXPERIMENTS

|  |  |
| --- | --- |
| ***Ex.***  ***No.*** | ***Name of the Experiment*** |
| **1.a.** | Perform encryption, decryption using the following substitution techniques   1. Ceaser cipher 2. Playfair cipher |
| **1.b.** | Perform encryption and decryption using following transposition techniques   1. Rail fence 2. Row & Column Transformation |
| **2.** | Apply DES algorithm for practical applications. |
| **3.** | Implement RSA Algorithm using HTML and JavaScript |
| **4.** | Implement the Diffie-Hellman Key Exchange algorithm for a given problem. |
| **5.** | Calculate the message digest of a text using the SHA-1 algorithm. |
| **6.** | Demonstrate intrusion detection system (ids) using any tool eg. Snort or any  other s/w. |

**Software Download Links:**

* **Visual Studio Code:** [**https://code.visualstudio.com/download**](https://code.visualstudio.com/download)
* **Snort -** [**https://www.snort.org/downloads**](https://www.snort.org/downloads)
* **N-Stalker -** [**https://www.nstalker.com/products/editions/free/download/**](https://www.nstalker.com/products/editions/free/download/)
* **GMER -** [**http://www.gmer.net/**](http://www.gmer.net/)
* **JAVA -** [**https://www.java.com/en/download/**](https://www.java.com/en/download/)

|  |  |
| --- | --- |
| **Ex. No : 1(a) Date :** | **Encryption and Decryption Using Ceaser Cipher** |

**AIM:**

To encrypt and decrypt the given message by using Ceaser Cipher

encryption algorithm.

# ALGORITHMS:

1. In Ceaser Cipher each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet.
2. For example, with a **left shift of 3**, **D** would be replaced by **A**, **E** would become **B**, and so on.
3. The encryption can also be represented using modular arithmetic by first transforming the letters into numbers, according to the scheme, **A = 0, B = 1, Z = 25.**
4. Encryption of a letter x by a shift n can be described mathematically as,

## En(x) = (x + n) mod26

1. Decryption is performed similarly,

***Dn (x)=(x - n) mod26***

# PROGRAM:

## CaesarCipher.java

class caesarCipher {

public static String encode(String enc, int offset) { offset = offset % 26 + 26;

StringBuilder encoded = new StringBuilder(); for (char i : enc.toCharArray()) {

if (Character.isLetter(i)) {

if (Character.isUpperCase(i)) {

encoded.append((char) ('A' + (i - 'A' + offset) % 26));

} else {

encoded.append((char) ('a' + (i - 'a' + offset) % 26));

}

} else {

encoded.append(i);

}

}

return encoded.toString();

}

public static String decode(String enc, int offset) { return encode(enc, 26 - offset);

}

public static void main(String[] args) throws java.lang.Exception { String msg = "Anna University";

System.out.println("Simulating Caesar Cipher\n ");

System.out.println("Input : " + msg); System.out.printf("Encrypted Message : "); System.out.println(caesarCipher.encode(msg, 3)); System.out.printf("Decrypted Message : ");

System.out.println(caesarCipher.decode(caesarCipher.encode(msg, 3), 3));

}

}

# OUTPUT:

Simulating Caesar Cipher

Input : Anna University

Encrypted Message : Dqqd Xqlyhuvlwb Decrypted Message : Anna University

# RESULT:

Thus the program for ceaser cipher encryption and decryption algorithm has been implemented and the output verified successfully.

|  |  |
| --- | --- |
| **Ex. No : 1(a) Date :** | **Playfair Cipher** |

# AIM:

To implement a program to encrypt a plain text and decrypt a cipher text

using play fair Cipher substitution technique.

# ALGORITHM:

1. To encrypt a message, one would break the message into digrams (groups of 2 letters)
2. For example, "HelloWorld" becomes "HE LL OW OR LD".
3. These digrams will be substituted using the key table.
4. Since encryption requires pairs of letters, messages with an odd number of characters usually append an uncommon letter, such as "X", to complete the final digram.
5. The two letters of the digram are considered opposite corners of a rectangle in the key table. To perform the substitution, apply the following 4 rules, in order, to each pair of letters in the plaintext:

# PROGRAM:

## playfairCipher.java

import java.awt.Point;

class playfairCipher {

private static char[][] charTable; private static Point[] positions;

private static String prepareText(String s, boolean chgJtoI) { s = s.toUpperCase().replaceAll("[^A-Z]", "");

return chgJtoI ? s.replace("J", "I") : s.replace("Q", "");

}

private static void createTbl(String key, boolean chgJtoI) { charTable = new char[5][5];

positions = new Point[26];

String s = prepareText(key + "ABCDEFGHIJKLMNOPQRSTUVWXYZ", chgJtoI);

int len = s.length();

for (int i = 0, k = 0; i < len; i++) { char c = s.charAt(i);

if (positions[c - 'A'] == null) { charTable[k / 5][k % 5] = c;

positions[c - 'A'] = new Point(k % 5, k / 5); k++;

}

}

}

private static String codec(StringBuilder txt, int dir) { int len = txt.length();

for (int i = 0; i < len; i += 2) { char a = txt.charAt(i);

char b = txt.charAt(i + 1);

int row1 = positions[a - 'A'].y; int row2 = positions[b - 'A'].y; int col1 = positions[a - 'A'].x; int col2 = positions[b - 'A'].x; if (row1 == row2) {

col1 = (col1 + dir) % 5; col2 = (col2 + dir) % 5;

} else if (col1 == col2) {

row1 = (row1 + dir) % 5; row2 = (row2 + dir) % 5;

} else {

int tmp = col1; col1 = col2; col2 = tmp;

}

txt.setCharAt(i, charTable[row1][col1]); txt.setCharAt(i + 1, charTable[row2][col2]);

}

return txt.toString();

}

private static String encode(String s) { StringBuilder sb = new StringBuilder(s); for (int i = 0; i < sb.length(); i += 2) {

if (i == sb.length() - 1) {

sb.append(sb.length() % 2 == 1 ? 'X' : "");

} else if (sb.charAt(i) == sb.charAt(i + 1)) { sb.insert(i + 1, 'X');

}

}

return codec(sb, 1);

}

private static String decode(String s) { return codec(new StringBuilder(s), 4);

}

public static void main(String[] args) throws java.lang.Exception { String key = "CSE";

String txt = "Security Lab"; /\* make sure string length is even \*/ /\* change J to I \*/

boolean chgJtoI = true; createTbl(key, chgJtoI);

String enc = encode(prepareText(txt, chgJtoI)); System.out.println("Simulating Playfair Cipher\n ");

System.out.println("Input Message : " + txt); System.out.println("\nEncrypted Message : " + enc); System.out.println("\nDecrypted Message : " + decode(enc));

System.out.println(“\nX denotes String ended");

}

}

# OUTPUT:

Simulating Playfair Cipher

Input Message : Security Lab

Encrypted Message : EABPUGYANSEZ Decrypted Message : SECURITYLABX

# RESULT:

Thus the program for playfair cipher encryption and decryption algorithm has been implemented and the output verified successfully.

|  |  |
| --- | --- |
| **Ex. No : 1(b) Date :** | **Rail Fence Cipher Transposition Technique** |

# AIM:

To implement a program for encryption and decryption using rail fence

transposition technique.

# ALGORITHM:

1. In the rail fence cipher, the plaintext is written downwards and diagonally on successive "rails" of an imaginary fence, then moving up when we reach the bottom rail.
2. When we reach the top rail, the message is written downwards again until the whole plaintext is written out.
3. The message is then read off in rows.

# PROGRAM:

## railFenceCipher.java

class railfenceCipherHelper { int depth;

String encode(String msg, int depth) throws Exception { int r = depth;

int l = msg.length(); int c = l / depth;

int k = 0;

char mat[][] = new char[r][c]; String enc = "";

for (int i = 0; i < c; i++) { for (int j = 0; j < r; j++) {

if (k != l) {

mat[j][i] = msg.charAt(k++);

} else {

mat[j][i] = 'X';

}

}

}

for (int i = 0; i < r; i++) { for (int j = 0; j < c; j++) {

enc += mat[i][j];

}

}

return enc;

}

String decode(String encmsg, int depth) throws Exception { int r = depth;

int l = encmsg.length(); int c = l / depth;

int k = 0;

char mat[][] = new char[r][c]; String dec = "";

for (int i = 0; i < r; i++) { for (int j = 0; j < c; j++) {

mat[i][j] = encmsg.charAt(k++);

}

}

for (int i = 0; i < c; i++) { for (int j = 0; j < r; j++) {

dec += mat[j][i];

}

}

return dec;

}

}

class railFenceCipher {

public static void main(String[] args) throws java.lang.Exception { railfenceCipherHelper rf = new railfenceCipherHelper();

String msg, enc, dec;

msg = "Anna University, Chennai"; int depth = 2;

enc = rf.encode(msg, depth); dec = rf.decode(enc, depth);

System.out.println("Simulating Railfence Cipher\n ");

System.out.println("Input Message : " + msg); System.out.println("Encrypted Message : " + enc); System.out.printf("Decrypted Message : " + dec);

}

}

# OUTPUT:

Simulating Railfence Cipher

-

Input Message : Anna University, Chennai Encrypted Message : An nvriy hnanaUiest,Ceni Decrypted Message : Anna University, Chennai

# RESULT:

Thus the java program for Rail Fence Transposition Technique has been implemented and the output verified successfully.

|  |  |
| --- | --- |
| **Ex. No : 1(b) Date :** | **Row and Column Transformation Technique** |

# AIM:

To implement a program for encryption and decryption by using row and

column transformation technique.

# ALGORITHM:

1. Consider the plain text hello world, and let us apply the simple columnar transposition technique as shown below

|  |  |  |  |
| --- | --- | --- | --- |
| h | e | l | l |
| o | w | o | r |
| l | d |  |  |

1. The plain text characters are placed horizontally and the cipher text is created with vertical format as: **holewdlo lr**.
2. Now, the receiver has to use the same table to decrypt the cipher text to plain text.

# PROGRAM:

## TransCipher.java

import java.util.\*; class TransCipher {

public static void main(String args[]) { Scanner sc = new Scanner(System.in); System.out.println("Enter the plain text"); String pl = sc.nextLine();

sc.close(); String s = ""; int start = 0;

for (int i = 0; i < pl.length(); i++) { if (pl.charAt(i) == ' ') {

s = s + pl.substring(start, i); start = i + 1;

}

}

s = s + pl.substring(start);

System.out.print(s); System.out.println();

// end of space deletion

int k = s.length(); int l = 0;

int col = 4;

int row = s.length() / col;

char ch[][] = new char[row][col]; for (int i = 0; i < row; i++) {

for (int j = 0; j < col; j++) { if (l < k) {

ch[i][j] = s.charAt(l); l++;

} else {

ch[i][j] = '#';

}

}

}

// arranged in matrix

char trans[][] = new char[col][row]; for (int i = 0; i < row; i++) {

for (int j = 0; j < col; j++) { trans[j][i] = ch[i][j];

}

}

for (int i = 0; i < col; i++) { for (int j = 0; j < row; j++) {

System.out.print(trans[i][j]);

}

}

// display System.out.println();

}

}

# OUTPUT:

Enter the plain text Security Lab SecurityLab Sreictuy

# RESULT:

Thus the java program for Row and Column Transposition Technique has been implemented and the output verified successfully.

|  |  |
| --- | --- |
| **Ex. No : 2**  **Date :** | **Data Encryption Standard (DES) Algorithm (User Message Encryption )** |

# AIM:

To use Data Encryption Standard (DES) Algorithm for a practical

application like User Message Encryption.

# ALGORITHM:

1. Create a DES Key.
2. Create a Cipher instance from Cipher class, specify the following information and separated by a slash (/).
   1. Algorithm name
   2. Mode (optional)
   3. Padding scheme (optional)
3. Convert String into ***Byte[]*** array format.
4. Make Cipher in encrypt mode, and encrypt it with ***Cipher.doFinal()*** method.
5. Make Cipher in decrypt mode, and decrypt it with ***Cipher.doFinal()*** method.

# PROGRAM:

## DESExample.java

import javax.crypto.Cipher;

import javax.crypto.KeyGenerator;

import javax.crypto.SecretKey;

import java.util.Base64;

class DESExample {

Cipher ecipher;

Cipher dcipher;

DESExample(SecretKey key) throws Exception {

ecipher = Cipher.getInstance("DES");

dcipher = Cipher.getInstance("DES");

ecipher.init(Cipher.ENCRYPT\_MODE, key);

dcipher.init(Cipher.DECRYPT\_MODE, key);

}

public String encrypt(String str) throws Exception {

// Encode the string into bytes using utf-8

byte[] utf8 = str.getBytes("UTF8");

// Encrypt

byte[] enc = ecipher.doFinal(utf8);

// Encode bytes to base64 to get a string

return Base64.getEncoder().encodeToString(enc);

}

public String decrypt(String str) throws Exception {

// Decode base64 to get bytes

byte[] dec = Base64.getDecoder().decode(str);

byte[] utf8 = dcipher.doFinal(dec);

// Decode using utf-8

return new String(utf8, "UTF8");

}

public static void main(String[] argv) throws Exception {

final String secretText = "Hello Everyone";

System.out.println("SecretText: " + secretText+ “\n”);

SecretKey key = KeyGenerator.getInstance("DES").generateKey();

DESExample encrypter = new DESExample(key);

String encrypted = encrypter.encrypt(secretText);

System.out.println("\nEncrypted Value: " + encrypted+"\n");

String decrypted = encrypter.decrypt(encrypted);

System.out.println("\nDecrypted: " + decrypted);

}

}

# OUTPUT:

SecretText: Hello Everyone

Encrypted Value: KZAgxtzFYQQNHzSrUD1Y0A==

Decrypted: Hello Everyone

# RESULT:

Thus the java program for DES Algorithm has been implemented and the output verified successfully.

|  |  |
| --- | --- |
| **Ex. No : 3**  **Date :** | **RSA Algorithm** |

# AIM:

To implement RSA (Rivest–Shamir–Adleman) algorithm by using HTML

and Javascript.

# ALGORITHM:

1. Choose two prime number p and q
2. Compute the value of n and **p**
3. Find the value of ***e*** (public key)
4. Compute the value of ***d*** (private key) using gcd()
5. Do the encryption and decryption
   1. Encryption is given as,

## c = te mod n

* 1. Decryption is given as,

***t = cd mod n***

# PROGRAM:

## rsa.html

<html>

<head>

<title>RSA Encryption</title>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

</head>

<body>

<center>

<h1>RSA Algorithm</h1>

<h2>Implemented Using HTML & Javascript</h2>

<hr>

<table>

<tr>

<td>Enter First Prime Number:</td>

<td><input type="number" value="53" id="p"></td>

</tr>

<tr>

<td>Enter Second Prime Number:</td>

<td><input type="number" value="59" id="q"></p>

</td>

</tr>

<tr>

<td>Enter the Message(cipher text):<br>[A=1, B=2,...]</td>

<td><input type="number" value="89" id="msg"></p>

</td>

</tr>

<tr>

<td>Public Key:</td>

<td>

<p id="publickey"></p>

</td>

</tr>

<tr>

<td>Exponent:</td>

<td>

<p id="exponent"></p>

</td>

</tr>

<tr>

<td>Private Key:</td>

<td>

<p id="privatekey"></p>

</td>

</tr>

<tr>

<td>Cipher Text:</td>

<td>

<p id="ciphertext"></p>

</td>

</tr>

<tr>

<td><button onclick="RSA();">Apply RSA</button></td>

</tr>

</table>

</center>

</body>

<script type="text/javascript"> function RSA() {

var gcd, p, q, no, n, t, e, i, x;

gcd = function (a, b) { return (!b) ? a : gcd(b, a % b); }; p = document.getElementById('p').value;

q = document.getElementById('q').value;

no = document.getElementById('msg').value; n = p \* q;

t = (p - 1) \* (q - 1);

for (e = 2; e < t; e++) { if (gcd(e, t) == 1) {

break;

}

}

for (i = 0; i < 10; i++) { x = 1 + i \* t

if (x % e == 0) { d = x / e; break;

}

}

ctt = Math.pow(no, e).toFixed(0); ct = ctt % n;

dtt = Math.pow(ct, d).toFixed(0); dt = dtt % n;

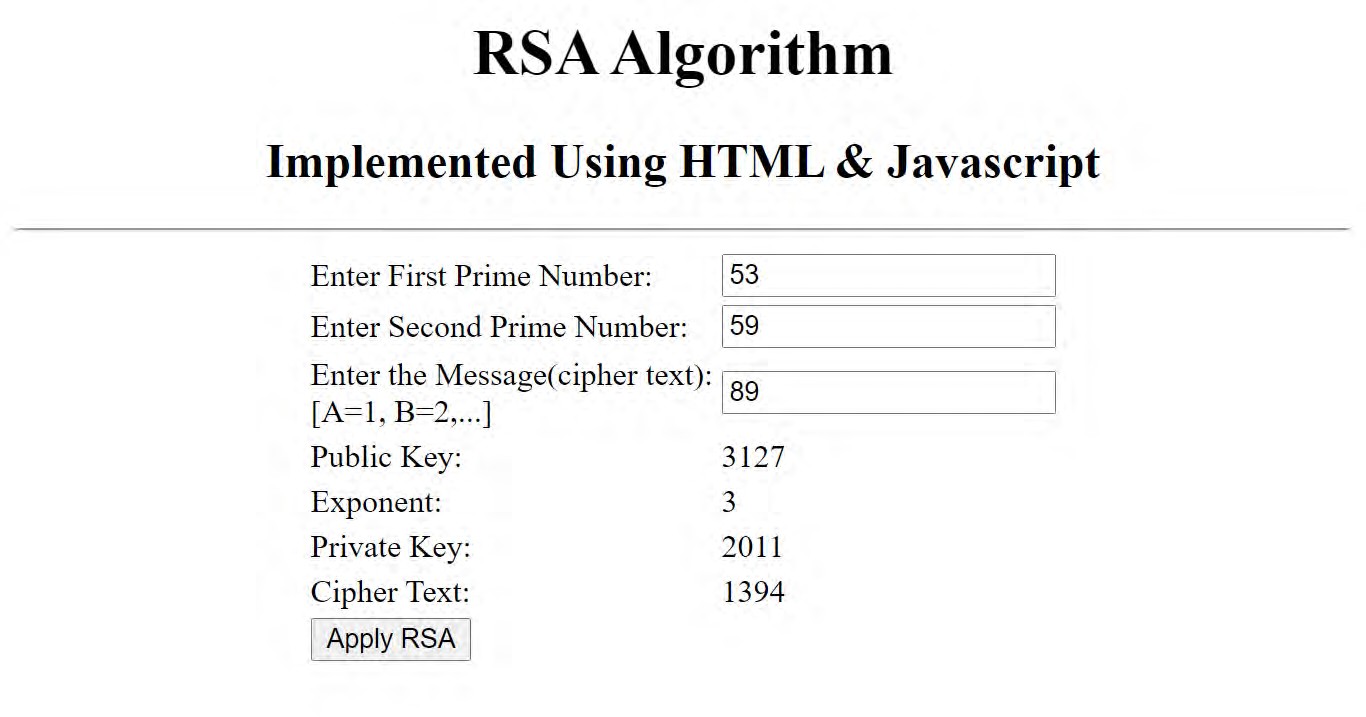
document.getElementById('publickey').innerHTML = n; document.getElementById('exponent').innerHTML = e; document.getElementById('privatekey').innerHTML = d; document.getElementById('ciphertext').innerHTML = ct;

}

</script>

</html>

# OUTPUT:



**RESULT:**

Thus the RSA algorithm has been implemented using HTML & CSS and the output has been verified successfully.

|  |  |
| --- | --- |
| **Ex. No : 4**  **Date :** | **Diffie-Hellman key exchange algorithm** |

# AIM:

To implement the Diffie-Hellman Key Exchange algorithm for a given

problem .

# ALGORITHM:

1. [Alice and Bob](https://en.wikipedia.org/wiki/Alice_and_Bob) publicly agree to use a modulus *p* = 23 and base *g* = 5 (which is a primitive root modulo 23).
2. Alice chooses a secret integer ***a*** = 4, then sends Bob *A* = *g****a*** mod *p*

o *A* = 5**4** mod 23 = 4

1. Bob chooses a secret integer ***b*** = 3, then sends Alice *B* = *g****b*** mod *p*

o *B* = 5**3** mod 23 = 10

1. Alice computes ***s*** = *B****a*** mod *p*

o ***s*** = 10**4** mod 23 = 18

1. Bob computes ***s*** = *A****b*** mod *p*

o ***s*** = 4**3** mod 23 = 18

1. Alice and Bob now share a secret (the number 18).

# PROGRAM:

## DiffieHellman.java

class DiffieHellman {

public static void main(String args[]) {

int p = 23; /\* publicly known (prime number) \*/ int g = 5; /\* publicly known (primitive root) \*/ int x = 4; /\* only Alice knows this secret \*/

int y = 3; /\* only Bob knows this secret \*/ double aliceSends = (Math.pow(g, x)) % p;

double bobComputes = (Math.pow(aliceSends, y)) % p; double bobSends = (Math.pow(g, y)) % p;

double aliceComputes = (Math.pow(bobSends, x)) % p; double sharedSecret = (Math.pow(g, (x \* y))) % p;

System.out.println("simulation of Diffie-Hellman key exchange algorithm\n”);

System.out.println("Alice Sends : " + aliceSends); System.out.println("Bob Computes : " + bobComputes); System.out.println("Bob Sends : " + bobSends);

System.out.println("Alice Computes : " + aliceComputes); System.out.println("Shared Secret : " + sharedSecret);

/\* shared secrets should match and equality is transitive \*/

if ((aliceComputes == sharedSecret) && (aliceComputes == bobComputes)) System.out.println("Success: Shared Secrets Matches! " + sharedSecret);

else

System.out.println("Error: Shared Secrets does not Match");

}

}

# OUTPUT:

simulation of Diffie-Hellman key exchange algorithm

Alice Sends : 4.0 Bob Computes : 18.0 Bob Sends : 10.0

Alice Computes : 18.0 Shared Secret : 18.0

Success: Shared Secrets Matches! 18.0

# RESULT:

Thus the *Diffie-Hellman key exchange algorithm* has been implemented using Java Program and the output has been verified successfully.

|  |  |
| --- | --- |
| **Ex. No : 5**  **Date :** | **SHA-1 Algorithm** |

# AIM:

To Calculate the message digest of a text using the SHA-1 algorithm.

# ALGORITHM:

1. Append Padding Bits
2. Append Length - 64 bits are appended to the end
3. Prepare Processing Functions
4. Prepare Processing Constants
5. Initialize Buffers
6. Processing Message in 512-bit blocks (L blocks in total message)

# PROGRAM:

## sha1.java

import java.security.\*;

public class sha1 {

public static void main(String[] a) { try {

MessageDigest md = MessageDigest.getInstance("SHA1"); System.out.println("Message digest object info:\n ");

System.out.println("Algorithm=" + md.getAlgorithm()); System.out.println("Provider=" + md.getProvider()); System.out.println("ToString=" + md.toString());

String input = ""; md.update(input.getBytes()); byte[] output = md.digest(); System.out.println();

System.out.println("SHA1(\"" + input + "\")=" + bytesToHex(output)); input = "abc";

md.update(input.getBytes()); output = md.digest(); System.out.println();

System.out.println("SHA1(\"" + input + "\")=" + bytesToHex(output)); input = "abcdefghijklmnopqrstuvwxyz";

md.update(input.getBytes());

output = md.digest(); System.out.println();

System.out.println("SHA1(\"" + input + "\")=" + bytesToHex(output)); System.out.println();

} catch (Exception e) { System.out.println("Exception:" + e);

}

}

private static String bytesToHex(byte[] b) {

char hexDigit[] = { '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F' };

StringBuffer buf = new StringBuffer();

for (byte aB : b) {

buf.append(hexDigit[(aB >> 4) & 0x0f]); buf.append(hexDigit[aB & 0x0f]);

}

return buf.toString();

}

}

# OUTPUT:

Message digest object info:

- Algorithm=SHA1

Provider=SUN version 12

ToString=SHA1 Message Digest from SUN, <initialized> SHA1("")=DA39A3EE5E6B4B0D3255BFEF95601890AFD80709 SHA1("abc")=A9993E364706816ABA3E25717850C26C9CD0D89D

SHA1("abcdefghijklmnopqrstuvwxyz")=32D10C7B8CF96570CA04CE37F2A19 D84240D3A89

# RESULT:

Thus the *Secure Hash Algorithm (SHA-1)* has been implemented and the output has been verified successfully.

|  |  |
| --- | --- |
| **Ex. No : 6**  **Date :** | **Demonstration of Intrusion Detection System(IDS)** |

# AIM:

To demonstrate Intrusion Detection System (IDS) using Snort software tool.

# STEPS ON CONFIGURING AND INTRUSION DETECTION:

1. Download Snort from the Snort.org website. (<http://www.snort.org/snort-> downloads)
2. Download Rules(https://[www.snort.org/snort-rules).](http://www.snort.org/snort-rules)) You must register to get the rules. (You should download these often)
3. Double click on the .exe to install snort. This will install snort in the “C:\Snort” folder.It is important to have WinPcap (https://[www.winpcap.org/install/)](http://www.winpcap.org/install/)) installed
4. Extract the Rules file. You will need WinRAR for the .gz file.
5. Copy all files from the “rules” folder of the extracted folder. Now paste the rules into *“C:\Snort\rules”* folder.
6. Copy “snort.conf” file from the “etc” folder of the extracted folder. You must paste it into “C:\Snort\etc” folder. Overwrite any existing file. Remember if you modify your snort.conf file and download a new file, you must modify it for Snort to work.
7. Open a command prompt (cmd.exe) and navigate to folder “C:\Snort\bin” folder. ( at the Prompt, type cd\snort\bin)
8. To start (execute) snort in sniffer mode use following command: snort -dev -i 3

-i indicates the interface number. You must pick the correct interface number. In my case, it is 3.

-dev is used to run snort to capture packets on your network.

To check the interface list, use following command: snort -W



Finding an interface

You can tell which interface to use by looking at the Index number and finding Microsoft. As you can see in the above example, the other interfaces are for VMWare. My interface is 3.

1. To run snort in IDS mode, you will need to configure the file “snort.conf” according to your network environment.
2. To specify the network address that you want to protect in snort.conf file, look for the following line.

var HOME\_NET 192.168.1.0/24 (You will normally see any here)

1. You may also want to set the addresses of DNS\_SERVERS, if you have some on your network.

Example:

example snort

1. Change the RULE\_PATH variable to the path of rules folder. var RULE\_PATH c:\snort\rules

path to rules

1. Change the path of all library files with the name and path on your system. and you must change the path of snort\_dynamicpreprocessorvariable. C:\Snort\lib\snort\_dynamiccpreprocessor

You need to do this to all library files in the “C:\Snort\lib” folder. The old path might be: “/usr/local/lib/…”. you will need to replace that path with your system path. Using C:\Snort\lib

1. Change the path of the “dynamicengine” variable value in the “snort.conf” file..

Example:

dynamicengine C:\Snort\lib\snort\_dynamicengine\sf\_engine.dll

15 Add the paths for “include classification.config” and “include reference.config” files.

include c:\snort\etc\classification.config include c:\snort\etc\reference.config

1. Remove the comment (#) on the line to allow ICMP rules, if it is commented with a #.

include $RULE\_PATH/icmp.rules

1. You can also remove the comment of ICMP-info rules comment, if it is commented.

include $RULE\_PATH/icmp-info.rules

1. To add log files to store alerts generated by snort, search for the “output log” test in snort.conf and add the following line:

output alert\_fast: snort-alerts.ids

1. Comment (add a #) the whitelist $WHITE\_LIST\_PATH/white\_list.rules and the blacklist

Change the nested\_ip inner , \ to nested\_ip inner #, \

1. Comment out (#) following lines: #preprocessor normalize\_ip4

#preprocessor normalize\_tcp: ips ecn stream #preprocessor normalize\_icmp4 #preprocessor normalize\_ip6

#preprocessor normalize\_icmp6

1. Save the “snort.conf” file.
2. To start snort in IDS mode, run the following command:

snort -c c:\snort\etc\snort.conf -l c:\snort\log -i 3 (Note: 3 is used for my interface card)

If a log is created, select the appropriate program to open it. You can use WordPard or NotePad++ to read the file.

To generate Log files in ASCII mode, you can use following command while running snort in IDS mode:

snort -A console -i3 -c c:\Snort\etc\snort.conf -l c:\Snort\log -K ascii

1. Scan the computer that is running snort from another computer by using PING or NMap (ZenMap).

After scanning or during the scan you can check the snort-alerts.ids file in the log folder to insure it is logging properly. You will see IP address folders appear.

Snort monitoring traffic –



# RESULT:

Thus the Intrusion Detection System(IDS) has been demonstrated by using the Open Source Snort Intrusion Detection Tool.