



Palestine Technical University (Kadoorie)
Faculty of Engineering and Technology
Department of Computer Systems Engineering

Cryptosystem and Network Security

- Triple S-DES System -



Prepared By:

Ahmad Haj-Qasem - 201810048

Mays Najjar - 201811598

Sondos Jarrar -201811910

Supervised by:

Dr. Anas Melhem

Tulkarm – Palestine

April.18th. 2022

Task definition :

In this project we have to develop a network application to test the 'Triple S-DESA'. for sure we need to implement this project to allow the encryption and decryption modes. So communicating parts must be represented as running on two separate computers connected via some channel.

Definition of an algorithm and mode of operation to be implemented:

Before starting understanding the code, let's check how the 'S-DES' works from the figure below :

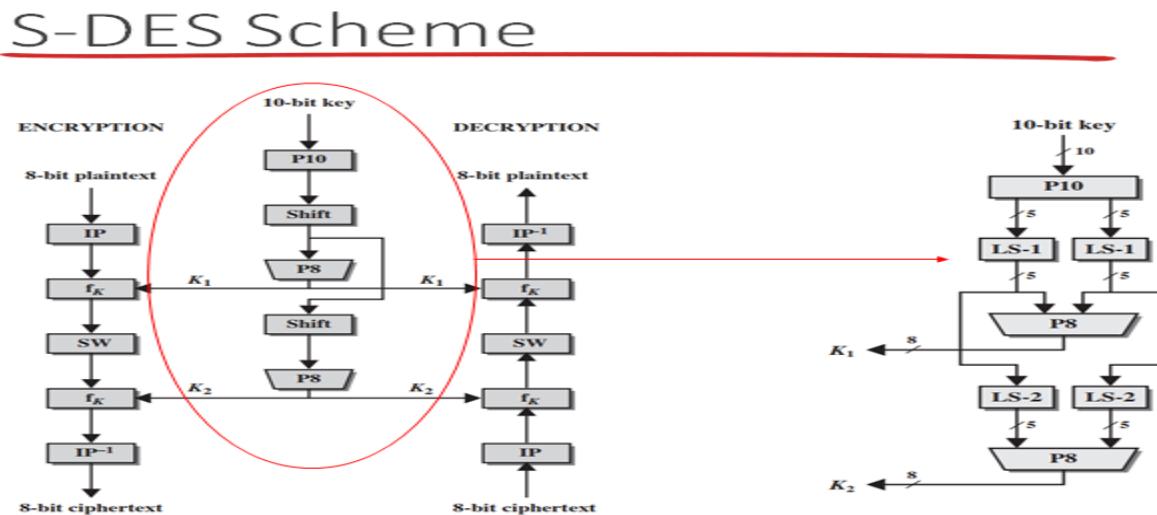


Figure 1: S-DES encryption and decryption modes

From the figure: 1 we can see the operations that we must be done before having the cipher text during the encryption, or the plain text in the decryption in S-DES . So first the 10-bits key enter P10 to do the permutation in a specific way and after that it enter a shift-1 operator which from this stage a copy from the key go to another shift-2 and P8 to wait for the key 1 be ready and enter together to the final function.

After the first shift for the key and to shrink it, it enters to P8 and becomes a 8-bits key, from this stage we can have our first key which enter to a specific function and then there's a swap happens between its bits and enters the function again with the second key.

The last step is to do IP^{-1} which is the inverse of the initial permutation and after this stage we have our cipher text.

But what we have to do is to implement the Triple S-DES, so we need to repeat some operations from the normal S-DES to do it in a more complex way.

Figure 2 shows the number of bits for the key and the plain text after the encryption for S-DES in a simple way :

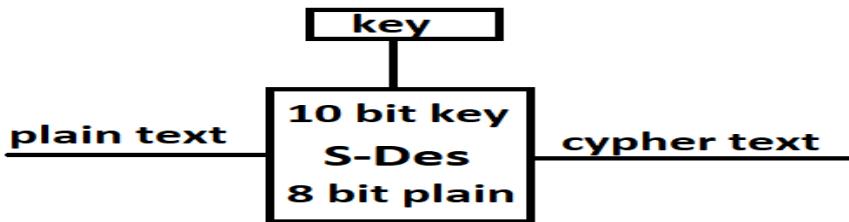


Figure 2: S-DES encryption

In the figure below we can notice how the Triple S-DES works in both, encryption and decryption modes .

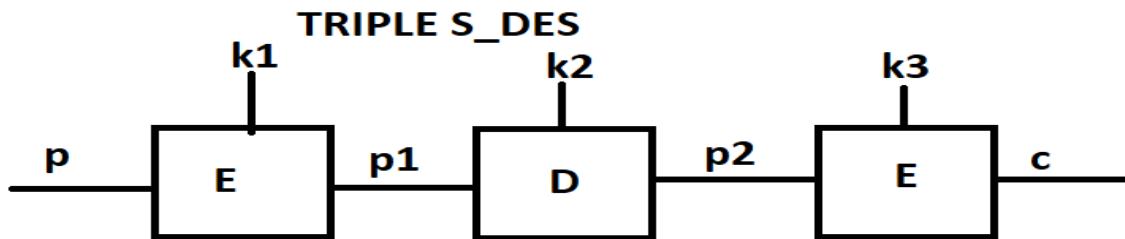


Figure 3: Triple S-DES encryption

As we can see from the figure above there's a special operation to do to have our cipher text which is : **Triple S-DES: $3S\text{-DES}(x) = E(K_3, (D(K_2 (E(K_1, x))))$**

We can notice that we need 3 keys with 56 bits for each, in order to have our cipher text and the same thing for the decryption operation to have the plain text as we can see from the figure 4 below.

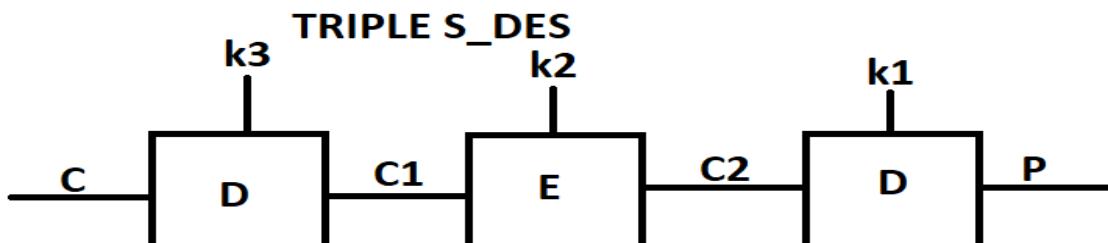


Figure 4: Triple S-DES decryption

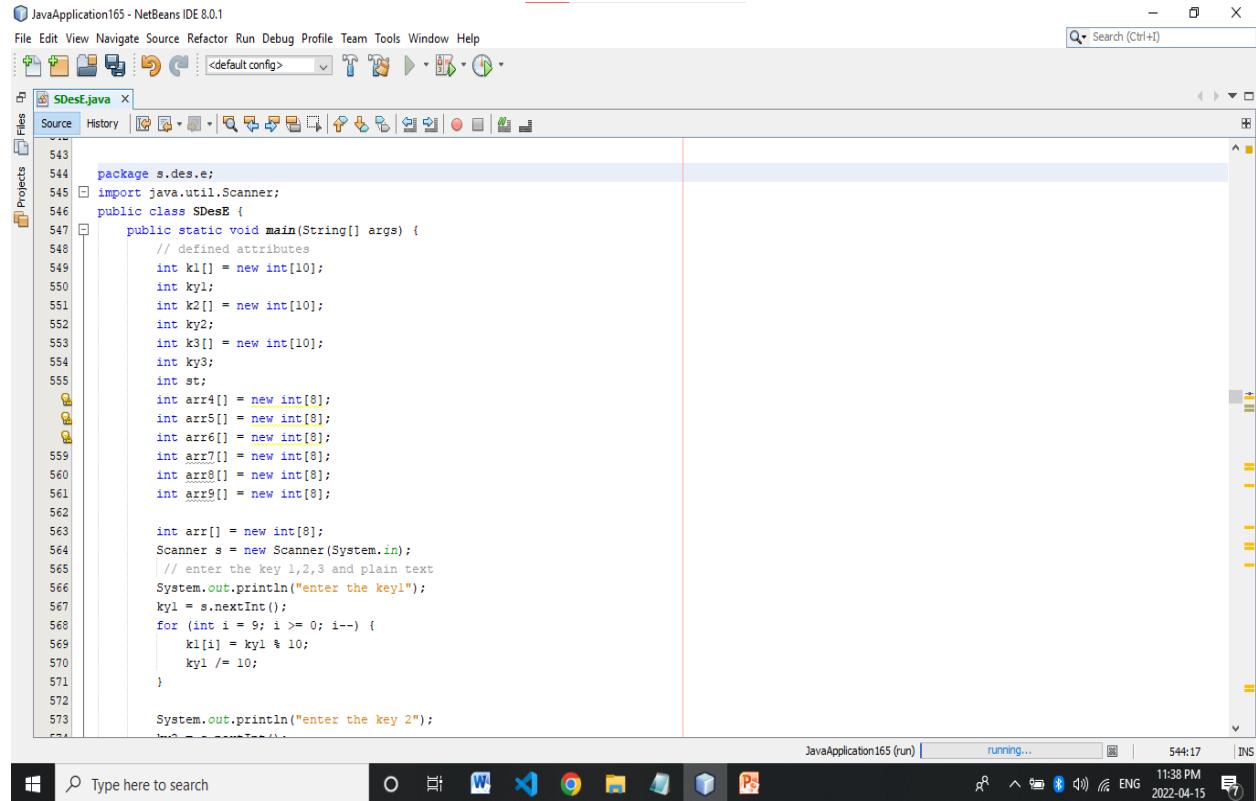
After all those steps all what we need is to make them work in two devices using a network with some codes to make it work and clear which we will see in some screenshots after a while in this project.

Description of developed application and its source codes:

Well, about the code , we wrote it from a to z ourselves without any source from the internet, and now we're gonna explain everything about the code and provide some screenshots to make it clear.

First of all let's start with the encryption operation to do $E(k_3, D(k_2, E(k_1, p)))$.

It starts with entering 8-bits plain text with 3 keys those we use in a different stages to do the triple S-Des .



The screenshot shows the NetBeans IDE interface with the following details:

- Title Bar:** JavaApplication165 - NetBeans IDE 8.0.1
- Toolbar:** File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
- Project Explorer:** Shows a single project named "JavaApplication165".
- Code Editor:** The file "SDesE.java" is open. The code implements the triple DES encryption process. It defines three 10x10 integer arrays (k1, k2, k3) and nine 8x1 integer arrays (arr4 through arr12). It uses a Scanner object to read three keys (k1, k2, k3) and plain text. The code then performs three rounds of DES encryption using the keys and the plain text.
- Status Bar:** Shows the application is running, the time is 11:38 PM, and the date is 2022-04-15.

```
543 package s.des.e;
544 import java.util.Scanner;
545 public class SDesE {
546     public static void main(String[] args) {
547         // defined attributes
548         int k1[] = new int[10];
549         int ky1;
550         int k2[] = new int[10];
551         int ky2;
552         int k3[] = new int[10];
553         int ky3;
554         int st;
555         int arr4[] = new int[8];
556         int arr5[] = new int[8];
557         int arr6[] = new int[8];
558         int arr7[] = new int[8];
559         int arr8[] = new int[8];
560         int arr9[] = new int[8];
561         int arr10[] = new int[8];
562         int arr11[] = new int[8];
563         int arr12[] = new int[8];
564         Scanner s = new Scanner(System.in);
565         // enter the key 1,2,3 and plain text
566         System.out.println("enter the key1");
567         ky1 = s.nextInt();
568         for (int i = 9; i >= 0; i--) {
569             k1[i] = ky1 % 10;
570             ky1 /= 10;
571         }
572         System.out.println("enter the key 2");
573         System.out.println("enter the plain text");
574         st = s.nextInt();
575         arr4[0] = st;
576         arr5[0] = st;
577         arr6[0] = st;
578         arr7[0] = st;
579         arr8[0] = st;
580         arr9[0] = st;
581         arr10[0] = st;
582         arr11[0] = st;
583         arr12[0] = st;
584         SDesE.sDesE(k1, k2, k3, arr4, arr5, arr6, arr7, arr8, arr9, arr10, arr11, arr12, s);
585     }
586     public static void sDesE(int k1[], int k2[], int k3[], int arr4[], int arr5[], int arr6[], int arr7[], int arr8[], int arr9[], int arr10[], int arr11[], int arr12[], Scanner s) {
587         int t1, t2, t3, t4, t5, t6, t7, t8, t9, t10, t11, t12, t13, t14, t15, t16, t17, t18, t19, t20, t21, t22, t23, t24, t25, t26, t27, t28, t29, t30, t31, t32, t33, t34, t35, t36, t37, t38, t39, t40, t41, t42, t43, t44, t45, t46, t47, t48, t49, t50, t51, t52, t53, t54, t55, t56, t57, t58, t59, t60, t61, t62, t63, t64, t65, t66, t67, t68, t69, t70, t71, t72, t73, t74, t75, t76, t77, t78, t79, t80, t81, t82, t83, t84, t85, t86, t87, t88, t89, t90, t91, t92, t93, t94, t95, t96, t97, t98, t99, t100, t101, t102, t103, t104, t105, t106, t107, t108, t109, t110, t111, t112, t113, t114, t115, t116, t117, t118, t119, t120, t121, t122, t123, t124, t125, t126, t127, t128, t129, t130, t131, t132, t133, t134, t135, t136, t137, t138, t139, t140, t141, t142, t143, t144, t145, t146, t147, t148, t149, t150, t151, t152, t153, t154, t155, t156, t157, t158, t159, t160, t161, t162, t163, t164, t165, t166, t167, t168, t169, t170, t171, t172, t173, t174, t175, t176, t177, t178, t179, t180, t181, t182, t183, t184, t185, t186, t187, t188, t189, t190, t191, t192, t193, t194, t195, t196, t197, t198, t199, t200, t201, t202, t203, t204, t205, t206, t207, t208, t209, t210, t211, t212, t213, t214, t215, t216, t217, t218, t219, t220, t221, t222, t223, t224, t225, t226, t227, t228, t229, t230, t231, t232, t233, t234, t235, t236, t237, t238, t239, t240, t241, t242, t243, t244, t245, t246, t247, t248, t249, t250, t251, t252, t253, t254, t255, t256, t257, t258, t259, t260, t261, t262, t263, t264, t265, t266, t267, t268, t269, t270, t271, t272, t273, t274, t275, t276, t277, t278, t279, t280, t281, t282, t283, t284, t285, t286, t287, t288, t289, t290, t291, t292, t293, t294, t295, t296, t297, t298, t299, t300, t301, t302, t303, t304, t305, t306, t307, t308, t309, t310, t311, t312, t313, t314, t315, t316, t317, t318, t319, t320, t321, t322, t323, t324, t325, t326, t327, t328, t329, t330, t331, t332, t333, t334, t335, t336, t337, t338, t339, t340, t341, t342, t343, t344, t345, t346, t347, t348, t349, t350, t351, t352, t353, t354, t355, t356, t357, t358, t359, t360, t361, t362, t363, t364, t365, t366, t367, t368, t369, t370, t371, t372, t373, t374, t375, t376, t377, t378, t379, t380, t381, t382, t383, t384, t385, t386, t387, t388, t389, t390, t391, t392, t393, t394, t395, t396, t397, t398, t399, t400, t401, t402, t403, t404, t405, t406, t407, t408, t409, t410, t411, t412, t413, t414, t415, t416, t417, t418, t419, t420, t421, t422, t423, t424, t425, t426, t427, t428, t429, t430, t431, t432, t433, t434, t435, t436, t437, t438, t439, t440, t441, t442, t443, t444, t445, t446, t447, t448, t449, t450, t451, t452, t453, t454, t455, t456, t457, t458, t459, t460, t461, t462, t463, t464, t465, t466, t467, t468, t469, t470, t471, t472, t473, t474, t475, t476, t477, t478, t479, t480, t481, t482, t483, t484, t485, t486, t487, t488, t489, t490, t491, t492, t493, t494, t495, t496, t497, t498, t499, t500, t501, t502, t503, t504, t505, t506, t507, t508, t509, t510, t511, t512, t513, t514, t515, t516, t517, t518, t519, t520, t521, t522, t523, t524, t525, t526, t527, t528, t529, t530, t531, t532, t533, t534, t535, t536, t537, t538, t539, t540, t541, t542, t543, t544, t545, t546, t547, t548, t549, t550, t551, t552, t553, t554, t555, t556, t557, t558, t559, t560, t561, t562, t563, t564, t565, t566, t567, t568, t569, t570, t571, t572, t573, t574, t575, t576, t577, t578, t579, t580, t581, t582, t583, t584, t585, t586, t587, t588, t589, t589, t590, t591, t592, t593, t594, t595, t596, t597, t598, t599, t599, t600, t601, t602, t603, t604, t605, t606, t607, t608, t609, t609, t610, t611, t612, t613, t614, t615, t616, t617, t618, t619, t619, t620, t621, t622, t623, t624, t625, t626, t627, t628, t629, t629, t630, t631, t632, t633, t634, t635, t636, t637, t638, t639, t639, t640, t641, t642, t643, t644, t645, t646, t647, t648, t649, t649, t650, t651, t652, t653, t654, t655, t656, t657, t658, t659, t659, t660, t661, t662, t663, t664, t665, t666, t667, t668, t669, t669, t670, t671, t672, t673, t674, t675, t676, t677, t678, t679, t679, t680, t681, t682, t683, t684, t685, t686, t687, t688, t689, t689, t690, t691, t692, t693, t694, t695, t696, t697, t698, t699, t699, t700, t701, t702, t703, t704, t705, t706, t707, t708, t709, t709, t710, t711, t712, t713, t714, t715, t716, t717, t718, t719, t719, t720, t721, t722, t723, t724, t725, t726, t727, t728, t729, t729, t730, t731, t732, t733, t734, t735, t736, t737, t738, t739, t739, t740, t741, t742, t743, t744, t745, t746, t747, t748, t749, t749, t750, t751, t752, t753, t754, t755, t756, t757, t758, t759, t759, t760, t761, t762, t763, t764, t765, t766, t767, t768, t769, t769, t770, t771, t772, t773, t774, t775, t776, t777, t778, t779, t779, t780, t781, t782, t783, t784, t785, t786, t787, t788, t789, t789, t790, t791, t792, t793, t794, t795, t796, t797, t798, t799, t799, t800, t801, t802, t803, t804, t805, t806, t807, t808, t809, t809, t810, t811, t812, t813, t814, t815, t816, t817, t818, t819, t819, t820, t821, t822, t823, t824, t825, t826, t827, t828, t829, t829, t830, t831, t832, t833, t834, t835, t836, t837, t838, t839, t839, t840, t841, t842, t843, t844, t845, t846, t847, t848, t849, t849, t850, t851, t852, t853, t854, t855, t856, t857, t858, t859, t859, t860, t861, t862, t863, t864, t865, t866, t867, t868, t869, t869, t870, t871, t872, t873, t874, t875, t876, t877, t878, t879, t879, t880, t881, t882, t883, t884, t885, t886, t887, t888, t889, t889, t890, t891, t892, t893, t894, t895, t896, t897, t898, t899, t899, t900, t901, t902, t903, t904, t905, t906, t907, t908, t909, t909, t910, t911, t912, t913, t914, t915, t916, t917, t918, t919, t919, t920, t921, t922, t923, t924, t925, t926, t927, t928, t929, t929, t930, t931, t932, t933, t934, t935, t936, t937, t938, t939, t939, t940, t941, t942, t943, t944, t945, t946, t947, t948, t949, t949, t950, t951, t952, t953, t954, t955, t956, t957, t958, t959, t959, t960, t961, t962, t963, t964, t965, t966, t967, t968, t969, t969, t970, t971, t972, t973, t974, t975, t976, t977, t978, t979, t979, t980, t981, t982, t983, t984, t985, t986, t987, t988, t989, t989, t990, t991, t992, t993, t994, t995, t996, t997, t998, t999, t999, t1000, t1001, t1002, t1003, t1004, t1005, t1006, t1007, t1008, t1009, t1009, t1010, t1011, t1012, t1013, t1014, t1015, t1016, t1017, t1018, t1019, t1019, t1020, t1021, t1022, t1023, t1024, t1025, t1026, t1027, t1028, t1029, t1029, t1030, t1031, t1032, t1033, t1034, t1035, t1036, t1037, t1038, t1039, t1039, t1040, t1041, t1042, t1043, t1044, t1045, t1046, t1047, t1048, t1049, t1049, t1050, t1051, t1052, t1053, t1054, t1055, t1056, t1057, t1058, t1059, t1059, t1060, t1061, t1062, t1063, t1064, t1065, t1066, t1067, t1068, t1069, t1069, t1070, t1071, t1072, t1073, t1074, t1075, t1076, t1077, t1078, t1079, t1079, t1080, t1081, t1082, t1083, t1084, t1085, t1086, t1087, t1088, t1089, t1089, t1090, t1091, t1092, t1093, t1094, t1095, t1096, t1097, t1098, t1099, t1099, t1100, t1101, t1102, t1103, t1104, t1105, t1106, t1107, t1108, t1109, t1109, t1110, t1111, t1112, t1113, t1114, t1115, t1116, t1117, t1118, t1119, t1119, t1120, t1121, t1122, t1123, t1124, t1125, t1126, t1127, t1128, t1129, t1129, t1130, t1131, t1132, t1133, t1134, t1135, t1136, t1137, t1138, t1139, t1139, t1140, t1141, t1142, t1143, t1144, t1145, t1146, t1147, t1148, t1149, t1149, t1150, t1151, t1152, t1153, t1154, t1155, t1156, t1157, t1158, t1159, t1159, t1160, t1161, t1162, t1163, t1164, t1165, t1166, t1167, t1168, t1169, t1169, t1170, t1171, t1172, t1173, t1174, t1175, t1176, t1177, t1178, t1179, t1179, t1180, t1181, t1182, t1183, t1184, t1185, t1186, t1187, t1188, t1189, t1189, t1190, t1191, t1192, t1193, t1194, t1195, t1196, t1197, t1198, t1198, t1199, t1199, t1200, t1201, t1202, t1203, t1204, t1205, t1206, t1207, t1208, t1209, t1209, t1210, t1211, t1212, t1213, t1214, t1215, t1216, t1217, t1218, t1219, t1219, t1220, t1221, t1222, t1223, t1224, t1225, t1226, t1227, t1228, t1229, t1229, t1230, t1231, t1232, t1233, t1234, t1235, t1236, t1237, t1238, t1239, t1239, t1240, t1241, t1242, t1243, t1244, t1245, t1246, t1247, t1248, t1249, t1249, t1250, t1251, t1252, t1253, t1254, t1255, t1256, t1257, t1258, t1259, t1259, t1260, t1261, t1262, t1263, t1264, t1265, t1266, t1267, t1268, t1269, t1269, t1270, t1271, t1272, t1273, t1274, t1275, t1276, t1277, t1278, t1279, t1279, t1280, t1281, t1282, t1283, t1284, t1285, t1286, t1287, t1288, t1289, t1289, t1290, t1291, t1292, t1293, t1294, t1295, t1296, t1297, t1298, t1298, t1299, t1299, t1300, t1301, t1302, t1303, t1304, t1305, t1306, t1307, t1308, t1309, t1309, t1310, t1311, t1312, t1313, t1314, t1315, t1316, t1317, t1318, t1319, t1319, t1320, t1321, t1322, t1323, t1324, t1325, t1326, t1327, t1328, t1329, t1329, t1330, t1331, t1332, t1333, t1334, t1335, t1336, t1337, t1338, t1339, t1339, t1340, t1341, t1342, t1343, t1344, t1345, t1346, t1347, t1348, t1349, t1349, t1350, t1351, t1352, t1353, t1354, t1355, t1356, t1357, t1358, t1359, t1359, t1360, t1361, t1362, t1363, t1364, t1365, t1366, t1367, t1368, t1369, t1369, t1370, t1371, t1372, t1373, t1374, t1375, t1376, t1377, t1378, t1379, t1379, t1380, t1381, t1382, t1383, t1384, t1385, t1386, t1387, t1388, t1389, t1389, t1390, t1391, t1392, t1393, t1394, t1395, t1396, t1397, t1398, t1398, t1399, t1399, t1400, t1401, t1402, t1403, t1404, t1405, t1406, t1407, t1408, t1409, t1409, t1410, t1411, t1412, t1413, t1414, t1415, t1416, t1417, t1418, t1419, t1419, t1420, t1421, t1422, t1423, t1424, t1425, t1426, t1427, t1428, t1429, t1429, t1430, t1431, t1432, t1433, t1434, t1435, t1436, t1437, t1438, t1439, t1439, t1440, t1441, t1442, t1443, t1444, t1445, t1446, t1447, t1448, t1449, t1449, t1450, t1451, t1452, t1453, t1454, t1455, t1456, t1457, t1458, t1459, t1459, t1460, t1461, t1462, t1463, t1464, t1465, t1466, t1467, t1468, t1469, t1469, t1470, t1471, t1472, t1473, t1474, t1475, t1476, t1477, t1478, t1479, t1479, t1480, t1481, t1482, t1483, t1484, t1485, t1486, t1487, t1488, t1489, t1489, t1490, t1491, t1492, t1493, t1494, t1495, t1496, t1497, t1498, t1498, t1499, t1499, t1500, t1501, t1502, t1503, t1504, t1505, t1506, t1507, t1508, t1509, t1509, t1510, t1511, t1512, t1513, t1514, t1515, t1516, t1517, t1518, t1519, t1519, t1520, t1521, t1522, t1523, t1524, t1525, t1526, t1527, t1528, t1529, t1529, t1530, t1531, t1532, t1533, t1534, t1535, t1536, t1537, t1538, t1539, t1539, t1540, t1541, t1542, t1543, t1544, t1545, t1546, t1547, t1548, t1549, t1549, t1550, t1551, t1552, t1553, t1554, t1555, t1556, t1557, t1558, t1559, t1559, t1560, t1561, t1562, t1563, t1564, t1565, t1566, t1567, t1568, t1569, t1569, t1570, t1571, t1572, t1573, t1574, t1575, t1576, t1577, t1578, t1579, t1579, t1580, t1581, t1582, t1583, t1584, t1585, t1586, t1587, t1588, t1589, t1589, t1590, t1591, t1592, t1593, t1594, t1595, t1596, t1597, t1598, t1598, t1599, t1599, t1600, t1601, t1602, t1603, t1604, t1605, t1606, t1607, t1608, t1609, t1609, t1610, t1611, t1612, t1613, t1614, t1615, t1616, t1617, t1618, t1619, t1619, t1620, t1621, t1622, t1623, t1624, t1625, t1626, t1627, t1628, t1629, t1629, t1630, t1631, t1632, t1633, t1634, t1635, t1636, t1637, t1638, t1639, t1639, t1640, t1641, t1642, t1643, t1644, t1645, t1646, t1647, t1648, t1649, t1649, t1650, t1651, t1652, t1653, t1654, t1655, t1656, t1657, t1658, t1659, t1659, t1660, t1661, t1662, t1663, t1664, t1665, t1666, t1667, t1668, t1669, t1669, t1670, t1671, t1672, t1673, t1674, t1675, t1676, t1677, t1678, t1679, t1679, t1680, t1681, t1682, t1683, t1684, t1685, t1686, t1687, t1688, t1689, t1689, t1690, t1691, t1692, t1693, t1694, t1695, t1696, t1697, t1698, t1698, t1699, t1699, t1700, t1701, t1702, t1703, t1704, t1705, t1706, t1707, t1708, t1709, t1709, t1710, t1711, t1712, t1713, t1714, t1715, t1716, t1717, t1718, t1719, t1719, t1720, t1721, t1722, t1723, t1724, t1725, t1726, t1727, t1728, t1729, t1729, t1730, t1731, t1732, t1733, t1734, t1735, t1736, t1737, t1738, t1739, t1739, t1740, t1741, t1742, t1743, t1744, t1745, t1746, t1747, t1748, t1749, t1749, t1750, t1751, t1752, t1753, t1754, t1755, t1756, t1757, t1758, t1759, t1759, t1760, t1761, t1762, t1763, t1764, t1765, t1766, t1767, t1768, t
```

```

564 Scanner s = new Scanner(System.in);
565 // enter the key 1,2,3 and plain text
566 System.out.println("enter the key1");
567 k1 = s.nextInt();
568 for (int i = 9; i >= 0; i--) {
569     k1[i] = k1 % 10;
570     k1 /= 10;
571 }
572
573 System.out.println("enter the key 2");
574 ky2 = s.nextInt();
575 for (int i = 9; i >= 0; i--) {
576     k2[i] = ky2 % 10;
577     ky2 /= 10;
578 }
579 System.out.println("enter the key 3");
580 ky3 = s.nextInt();
581 for (int i = 9; i >= 0; i--) {
582     k3[i] = ky3 % 10;
583     ky3 /= 10;
584 }
585
586 System.out.println("enter the plain text");
587 st = s.nextInt();
588 for (int i = 7; i >= 0; i--) {
589     arr1[i] = st % 10;
590     st = st / 10;
591 }
592
593 */
594 // make equation E(k3,D(k2,E(k1,p) triple s des
595 arr4 = enc(arr, k1);
596 System.out.println("the step 1 result");
597 for (int i = 0; i < 8; i++) {
598     System.out.print(arr4[i]);
599 }
600 System.out.println();
601 System.out.println(".....");
602
603 arr5 = dec(arr4, k2);
604 System.out.println("the step 2 result ");
605 for (int i = 0; i < 8; i++) {
606     System.out.print(arr5[i]);
607 }
608 System.out.println();
609 System.out.println(".....");
610 // the final result store in arr6
611 arr6 = enc(arr5, k3);
612 System.out.println("the cipher text and the final result ");
613 for (int i = 0; i < 8; i++) {
614     System.out.print(arr6[i]);
615 }
616 System.out.println();
617 System.out.println(".....");
618
619 //this method to make ip to plain text
620
621
622
623
624

```

The first stage is to encrypt the key1 with the plain text , then the next one is to decrypt the k2 with the output from the first stage , and then the last one is to encrypt the k3 with the output from the second stage as follow .

```

594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624

```

But we must know how each works, so let's explain every stage , the encryption and the decryption Proseccos.

First let's see how the encryption will work :

The screenshot shows the NetBeans IDE interface with the file 'SDES.java' open. The code implements the SDES encryption algorithm. It starts by generating keys (key1 and key2) from a provided key array. Then, it processes the plain text array through a series of operations including an initial permutation (IP), followed by two rounds of substitution and permutation (SW), and another IP permutation. The final output is printed to the console.

```
864  
865 // this method to make encryption to the plain text  
866 public static int[] enc(int arr[], int ke[]) {  
867  
868     int arr1[] = new int[8];  
869     int key1[] = new int[8];  
870     int key2[] = new int[8];  
871     int arr2[] = new int[8];  
872     int arr3[] = new int[8];  
873     int arr31[] = new int[4];  
874     int arr3r[] = new int[4];  
875     int arr4[] = new int[8];  
876     int keys[] = new int[16];  
877  
878     // this call key generator  
879     keys = gen(ke);  
880     // return array with 2 key each key 8 bit  
881     for (int i = 0; i < 8; i++) {  
882         key1[i] = keys[i];  
883         key2[i] = keys[i + 8];  
884     }  
885     for (int i = 0; i < 8; i++)  
886         arr1[i] = IP(arr, i);  
887  
888     // call fk to make operation to the plain text (1st section ) of sdes  
889     arr2 = fun(arr1, key1);  
890     System.out.println("the array after SW");  
891     for (int i = 0; i < 8; i++)  
892         System.out.print(arr2[i]);  
893     System.out.println();  
894  
895 }  
896  
897 }  
898
```

From the figure above we can see that the process starts with generating the keys (2) , but before continuing let's check the code of generating the 2 keys :

The screenshot shows the NetBeans IDE interface with the file 'SDES.java' open. The code implements the key generation process. It starts by reading a 10-bit key from standard input. This key is then processed through a P10 permutation to produce two 8-bit keys, key1 and key2. The code also includes a section for generating a 16-bit key, which is currently commented out.

```
926  
927 // this method to generate a key to make two keys each key with 8-bits  
928 public static int[] gen(int a[]) {  
929     Scanner s = new Scanner(System.in);  
930  
931     int keys[] = new int[16];  
932     int a1[] = new int[10];  
933     int k1[] = new int[8];  
934     int k2[] = new int[8];  
935     int a1l[] = new int[5];  
936     int a1r[] = new int[5];  
937     int a1lr[] = new int[5];  
938     int a1rr[] = new int[5];  
939  
940     System.out.println(" ur key");  
941     for (int i = 0; i < 10; i++)  
942         System.out.print(a1[i]);  
943     System.out.println();  
944     // call p10 method to make transposition  
945     for (int i = 0; i < 10; i++)  
946         a1[i] = pi0(a, i);  
947  
948     System.out.println("the key after p10");  
949     for (int i = 0; i < 10; i++)  
950         System.out.print(a1[i]);  
951     System.out.println();  
952  
953     for (int i = 0; i < 5; i++) {  
954         a1l[i] = a1[i];  
955         a1r[i] = a1[i + 5];  
956     }  
957     //make rotate to the key with value 1  
958 }
```

First a 10-bits key enters a P10 to make some permutation on it and then the output key from the p10 divided into 2 parts , left and right , as in the code above .

After that the two parts of the key rotate with 1-bit to the left then to the P8 which we will see its code soon, then after P8 , key 1 is ready ! which we can see in the code bellow.

The screenshot shows the NetBeans IDE interface with the title "JavaApplication165 - NetBeans IDE 8.0.1". The left sidebar includes a Projects tab and a Files tab. The main area displays the code for the SDesJava.java file. The code implements the DES encryption algorithm using S-boxes and P-boxes. It includes functions for key rotation, transposition, and substitution. The code uses arrays for state and key storage, and prints intermediate results to the console.

```
956
957
958
959
960
961     //make rotation to the key with value 1
962     aill[4] = aill[0];
963     aill[0] = aill[4];
964     for (int i = 0; i < 4; i++) {
965         aill[i] = aill[i + 1];
966         aill[i + 1] = airr[i];
967     }
968     for (int i = 0; i < 5; i++) {
969         ai[1] = aill[i];
970         ai[i + 5] = airr[i];
971 }
972 System.out.println("the key after rotation:");
973 for (int i = 0; i < 10; i++)
974     System.out.print(ai[i]);
975 System.out.println();
976 // call p8 to make transposition and substitution
977 // store key to kl array
978 for (int i = 0; i < 8; i++)
979     kl[i] = p8(ai, i);
980
981 System.out.println("the key l:");
982 for (int i = 0; i < 8; i++)
983     System.out.print(kl[i]);
984 System.out.println();
```

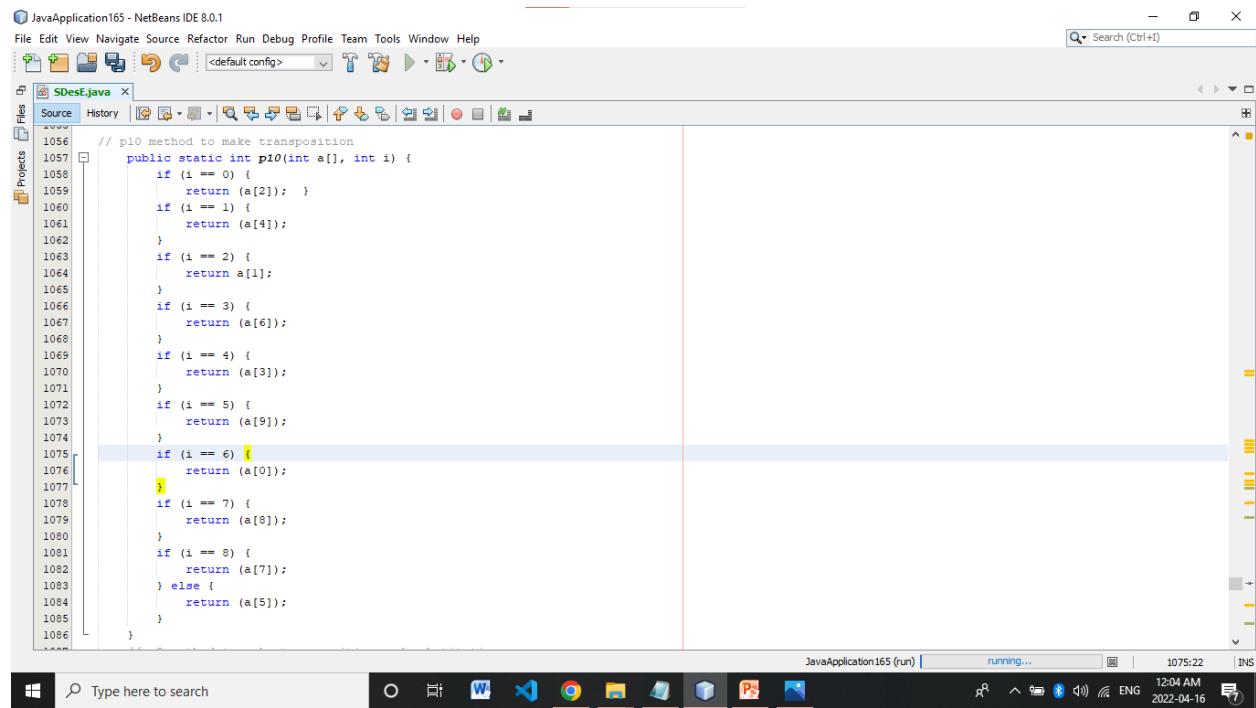
The P8 code to do permutation :

The screenshot shows the NetBeans IDE interface with the following details:

- Title Bar:** JavaApplication165 - NetBeans IDE 8.0.1
- Menu Bar:** File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
- Toolbar:** Includes icons for New Project, Open Project, Save, Undo, Redo, Cut, Copy, Paste, Find, Replace, and others.
- Search Bar:** Search (Ctrl+I)
- Project Explorer:** Shows a single project named "JavaApplication165".
- Files Tab:** Shows the current file is "SDest.java". Other tabs include "Source" and "History".
- Code Editor:** Displays Java code for a method named "p8". The code uses a series of if statements to return elements at indices 0 through 8 of an array "a".
- Status Bar:** Shows the application is running, the date and time (12:05 AM 2022-04-16), and system status (ENG).

```
1086 }
1087 }
1088 }
1089 // p8 method to make transposition and substitution
1090 public static int p8(int a[], int i) {
1091     if (i == 0) {
1092         return (a[5]);
1093     }
1094     if (i == 1) {
1095         return (a[2]);
1096     }
1097     if (i == 2) {
1098         return (a[6]);
1099     }
1100     if (i == 3) {
1101         return (a[3]);
1102     }
1103     if (i == 4) {
1104         return (a[7]);
1105     }
1106     if (i == 5) {
1107         return (a[4]);
1108     }
1109     if (i == 6) {
1110         return (a[9]);
1111     } else {
1112         return (a[8]);
1113     }
1114 }
1115 }
1116 }
1117 }
```

The P10 code which help in the first step with the origin key :

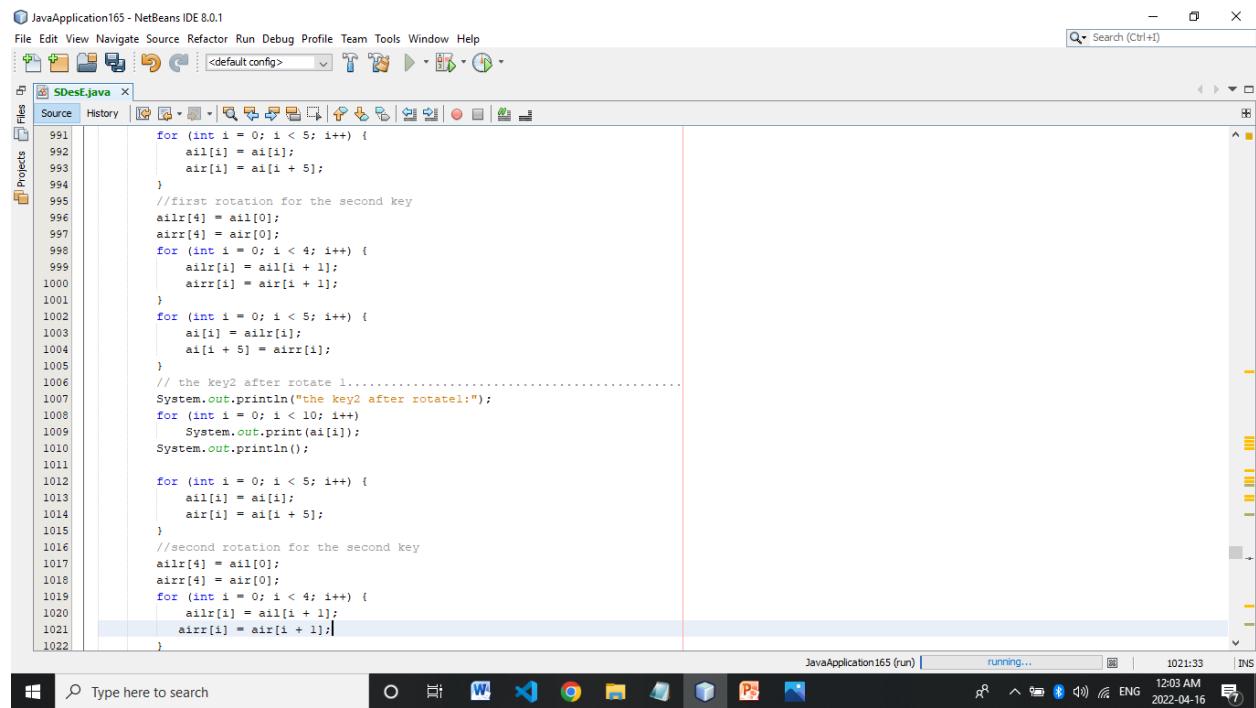


```

1056 // p10 method to make transposition
1057 public static int p10(int a[], int i) {
1058     if (i == 0) {
1059         return (a[2]);
1060     }
1061     if (i == 1) {
1062         return (a[4]);
1063     }
1064     if (i == 2) {
1065         return a[1];
1066     }
1067     if (i == 3) {
1068         return (a[6]);
1069     }
1070     if (i == 4) {
1071         return (a[3]);
1072     }
1073     if (i == 5) {
1074         return (a[9]);
1075     }
1076     if (i == 6) {
1077         return (a[0]);
1078     }
1079     if (i == 7) {
1080         return (a[8]);
1081     }
1082     if (i == 8) {
1083         return (a[7]);
1084     } else {
1085         return (a[5]);
1086     }
}

```

After that and to generate the key2 all what we need is to do another 2bits rotate to the left of the two parts that we did a 1-bit rotation in the beginning and then to enter them to the P8 as in the code bellow :



```

991     for (int i = 0; i < 5; i++) {
992         a1l[i] = ai1[i];
993         air[i] = ai1[i + 5];
994     }
995     //first rotation for the second key
996     a1lr[4] = a1l[0];
997     a1lr[4] = air[0];
998     for (int i = 0; i < 4; i++) {
999         a1lr[i] = a1l[i + 1];
1000         airr[i] = air[i + 1];
1001     }
1002     for (int i = 0; i < 5; i++) {
1003         ai1[i] = a1lr[i];
1004         ai1[i + 5] = airr[i];
1005     }
1006     // the key2 after rotate 1.....
1007     System.out.print("the key2 after rotate1:");
1008     for (int i = 0; i < 10; i++) {
1009         System.out.print(ai1[i]);
1010     }
1011     System.out.println();
1012     for (int i = 0; i < 5; i++) {
1013         a1l[i] = ai1[i];
1014         air[i] = ai1[i + 5];
1015     }
1016     //second rotation for the second key
1017     a1lr[4] = a1l[0];
1018     a1lr[4] = air[0];
1019     for (int i = 0; i < 4; i++) {
1020         a1lr[i] = a1l[i + 1];
1021         airr[i] = air[i + 1];
1022     }

```

The screenshot shows the NetBeans IDE interface with the file 'SDes.java' open. The code implements the S-Box function. It includes a main loop for generating key2, printing it, and then generating keys from k1 and k2. The code uses System.out.println statements to output intermediate values.

```
1023
1024
1025
1026     for (int i = 0; i < 5; i++) {
1027         ai[i] = a1r[i];
1028         ai[i + 5] = a1r[i];
1029     }
1030     // the key2 after rotate two .....
1031     System.out.println("the key2 after rotate2");
1032     for (int i = 0; i < 10; i++)
1033         System.out.print(ai[i]);
1034
1035     System.out.println();
1036     for (int i = 0; i < 8; i++)
1037         k2[i] = p8(ai, i);
1038
1039     System.out.println("the key two: ");
1040     for (int i = 0; i < 8; i++)
1041         System.out.print(k2[i]);
1042     System.out.println();
1043
1044
1045     for (int i = 0; i < 8; i++) {
1046         keys[i] = k1[i];
1047         keys[i + 8] = k2[i];
1048     }
1049
1050     return keys;
1051 }
1052
1053 }
```

Now let's continue with our encryption process :

The 8-bit plain text enters the stage then it enters –IP- which is the initial permutation so let's see its code :

The screenshot shows the NetBeans IDE interface with the file 'SDes.java' open. The code defines a method 'IP' for performing an initial permutation on an array of integers. The permutation follows a specific transposition pattern where indices 0 through 7 map to new positions 2, 6, 3, 7, 4, 5, 0, 1 respectively.

```
621
622
623 //this method to make IP to plain text
624 public static int IP(int arr[], int i) {
625     if (i == 0) {
626         return (arr[1]); /* transposition */
627     }
628     if (i == 1) {
629         return (arr[5]);
630     }
631     if (i == 2) {
632         return (arr[2]);
633     }
634     if (i == 3) {
635         return (arr[0]);
636     }
637     if (i == 4) {
638         return (arr[3]);
639     }
640     if (i == 5) {
641         return (arr[7]);
642     }
643     if (i == 6) {
644         return (arr[4]);
645     } else {
646         return (arr[6]);
647     }
648 }
649
650 // this method make expanding to right part after the plain text exit IP method
651 }
```

The output from the IP goes to fk which is kindda function do some complex processes we will see from the code :

```

709 // this method use to control operations or u can say fk
710 public static int[] fun(int arr1[], int key[])
711 {
712     int arr2[] = new int[8]; // defined attributes
713     int l0[] = new int[4];
714     int r0[] = new int[4];
715     int l1[] = new int[4];
716     int r1[] = new int[4];
717     int re0[] = new int[8];
718     int rel[] = new int[8];
719     int s0[] = {{1, 0, 3, 2}, {3, 2, 1, 0}, {0, 2, 1, 3}, {3, 1, 3, 2}};
720     int s1[] = {{0, 1, 2, 3}, {2, 0, 1, 3}, {3, 0, 1, 0}, {2, 1, 0, 3}};
721     int s0c = 0;
722     int s0r = 0;
723     int s1c = 0;
724     int s1r = 0;
725     int s0e = 0;
726     int s1e = 0;
727     int sbox[] = new int[4];
728     System.out.println("after IP:");
729     for (int i = 0; i < 8; i++)
730     {
731         System.out.print(arr1[i]);
732     }
733
734     // this operation use to divide the plain text to two part left and right
735     for (int i = 0; i < 4; i++)
736     {
737         l0[i] = arr1[i];
738
739         for (int i = 0; i < 4; i++)
740             r0[i] = arr1[i + 4];
    }

```

This function receive key1 and the output from the IP, it divides it to 2 parts with 4bits for each , left and right as in the code above.

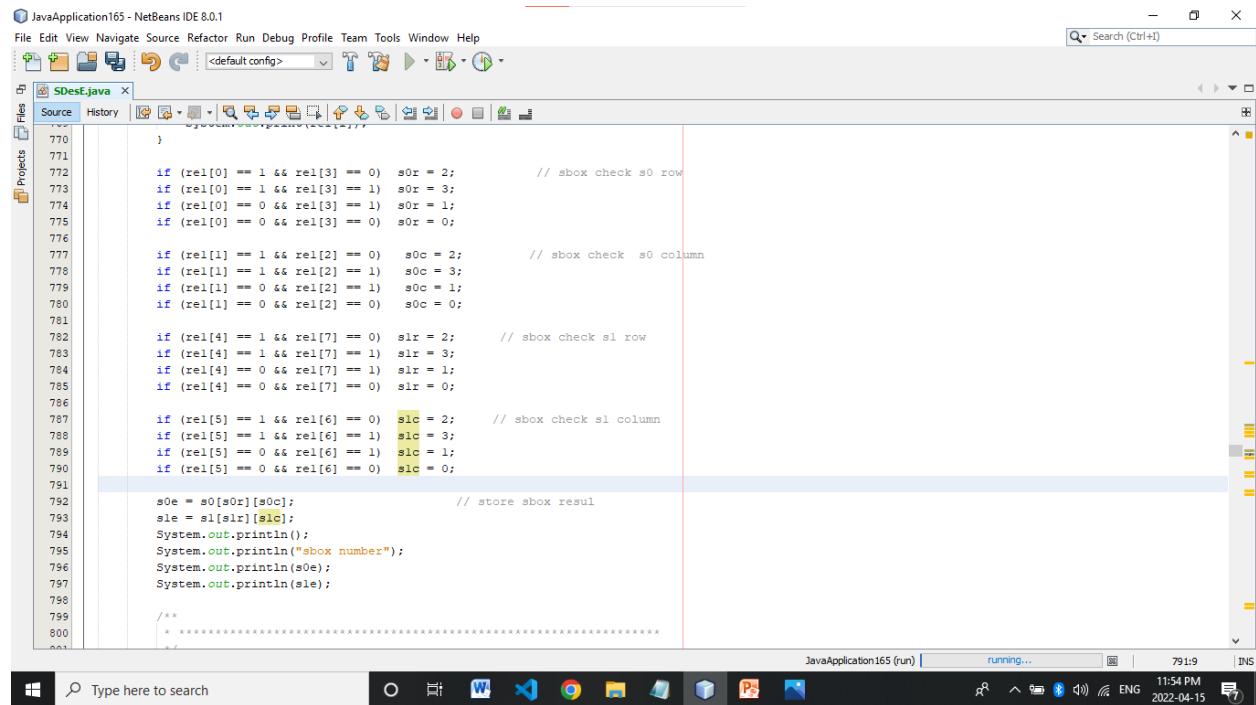
The right part enters to an expanding process to make it bits which we will show its code,
then the output from the expanding will enter with the key1 in an X-OR process as in the code ,

```

741     System.out.println("the left part:");
742     for (int i = 0; i < 4; i++)
743         System.out.print(l0[i]);
744     System.out.println();
745
746     System.out.println("the right part:");
747     for (int i = 0; i < 4; i++)
748         System.out.print(r0[i]);
749
750     // make an attribute to store the right part inside it :
751     for (int i = 0; i < 4; i++)
752         l1[i] = r0[i];
753     System.out.println();
754
755     System.out.println(" the expantion:");
756     // call method expand to make expantion to the right part
757     for (int i = 0; i < 8; i++)
758         re0[i] = expand(r0, i);
759     System.out.println("The right part after the expantion ");
760     for (int i = 0; i < 8; i++)
761         System.out.print(re0[i]);
762     System.out.println();
763
764     System.out.println("key xor re0:");
765     // make xor between the right part with the key which is the output of key generator
766     for (int i = 0; i < 8; i++)
767         rel[i] = key[i] ^ re0[i];
768     System.out.println("The output of the X-OR operation between the key and the right part:");
769     for (int i = 0; i < 8; i++)
770         System.out.print(rel[i]);
    }

```

Each part from the output enter to a process called S-Box to do a specific operation as in the code but the ouptut from the S-Box will be 2-bits instead of 4-bits as the input .



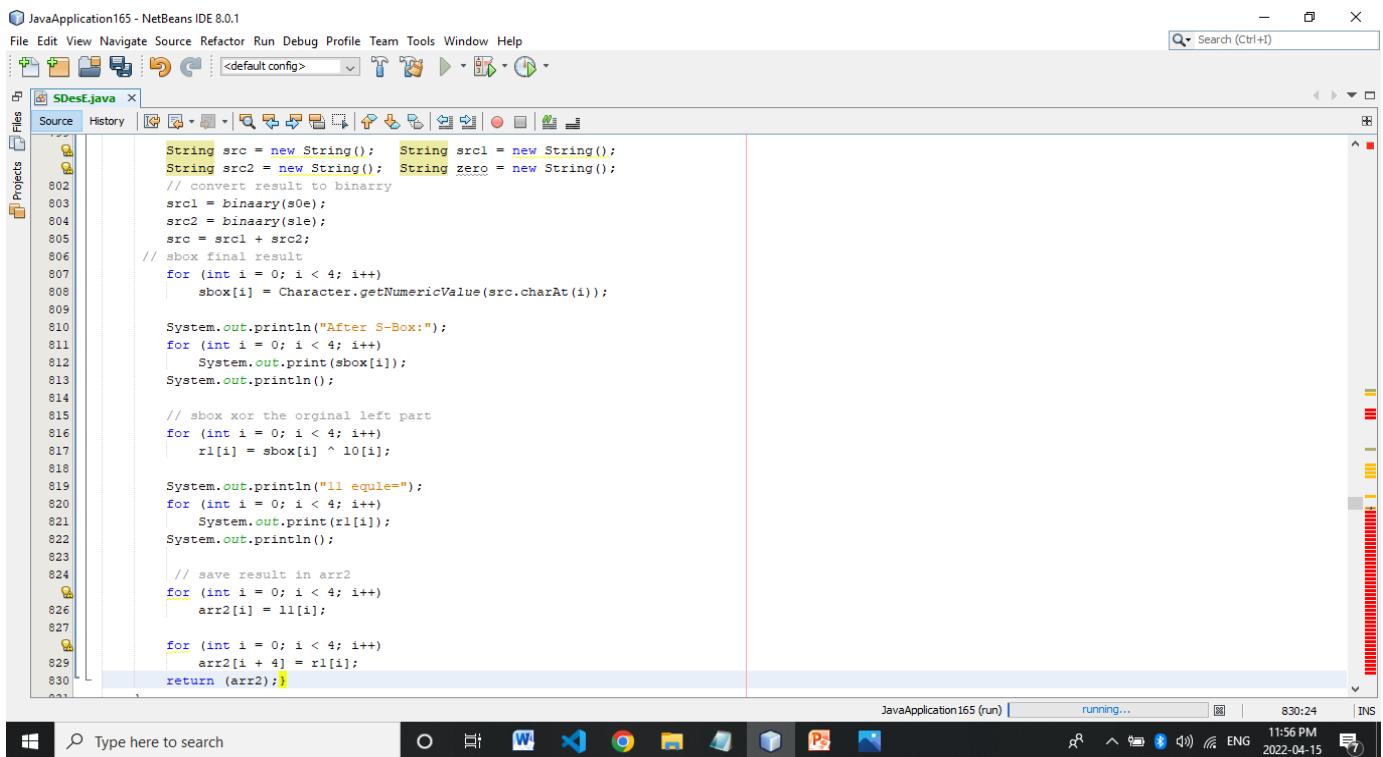
The screenshot shows the NetBeans IDE interface with the title "JavaApplication165 - NetBeans IDE 8.0.1". The code editor window displays Java code for an S-Box implementation. The code includes logic to map 4-bit inputs to 2-bit outputs (s0r, s0c, s1r, s1c) based on conditions involving `rel[0]` through `rel[7]`. It then stores the results in `s0e` and `s1e` variables. The code is annotated with comments explaining the operations.

```

770 }
771
772 if (rel[0] == 1 && rel[3] == 0) s0r = 2; // sbox check s0 row
773 if (rel[0] == 1 && rel[3] == 1) s0r = 3;
774 if (rel[0] == 0 && rel[3] == 1) s0r = 1;
775 if (rel[0] == 0 && rel[3] == 0) s0r = 0;
776
777 if (rel[1] == 1 && rel[2] == 0) s0c = 2; // sbox check s0 column
778 if (rel[1] == 1 && rel[2] == 1) s0c = 3;
779 if (rel[1] == 0 && rel[2] == 1) s0c = 1;
780 if (rel[1] == 0 && rel[2] == 0) s0c = 0;
781
782 if (rel[4] == 1 && rel[7] == 0) s1r = 2; // sbox check s1 row
783 if (rel[4] == 1 && rel[7] == 1) s1r = 3;
784 if (rel[4] == 0 && rel[7] == 1) s1r = 1;
785 if (rel[4] == 0 && rel[7] == 0) s1r = 0;
786
787 if (rel[5] == 1 && rel[6] == 0) s1c = 2; // sbox check s1 column
788 if (rel[5] == 1 && rel[6] == 1) s1c = 3;
789 if (rel[5] == 0 && rel[6] == 1) s1c = 1;
790 if (rel[5] == 0 && rel[6] == 0) s1c = 0;
791
792 s0e = s0[s0r][s0c]; // store sbox result
793 s1e = s1[s1r][s1c];
794 System.out.println();
795 System.out.println("sbox number");
796 System.out.println(s0e);
797 System.out.println(s1e);
798
799 /**
800 * ****
801 */
802

```

Then what will happen is X-Or between the first left part with the output of P4 which the input of it Is the output from the S-Box and the result will be with a swap .

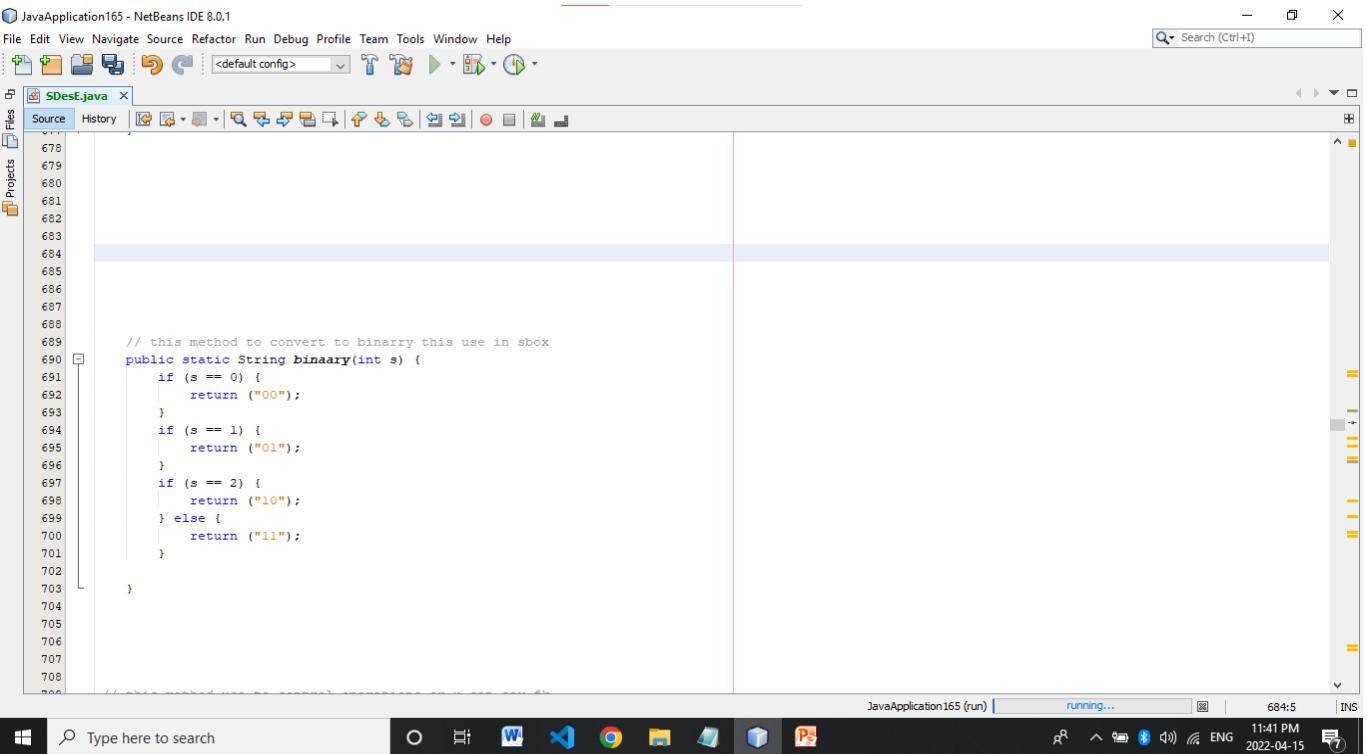


The screenshot shows the NetBeans IDE interface with the title "JavaApplication165 - NetBeans IDE 8.0.1". The code editor window displays Java code for an S-Box implementation. The code includes logic to convert binary strings to binary arrays, perform an S-Box transformation, and then perform an X-OR operation between the original left part and the S-Box result. The code is annotated with comments explaining the operations.

```

802 String src = new String(); String src1 = new String();
803 String src2 = new String(); String zero = new String();
804 // convert result to binary
805 src1 = binary(s0e);
806 src2 = binary(s1e);
807 src = src1 + src2;
808 // sbox final result
809 for (int i = 0; i < 4; i++)
810 | sbox[i] = Character.getNumericValue(src.charAt(i));
811
812 System.out.println("After S-Box:");
813 for (int i = 0; i < 4; i++)
814 | System.out.print(sbox[i]);
815 System.out.println();
816
817 // sbox xor the orginal left part
818 for (int i = 0; i < 4; i++)
819 | rl[i] = sbox[i] ^ 10[i];
820
821 System.out.println("11 equel=");
822 for (int i = 0; i < 4; i++)
823 | System.out.print(rl[i]);
824 System.out.println();
825
826 // save result in arr2
827 for (int i = 0; i < 4; i++)
828 | arr2[i] = 11[i];
829
830 for (int i = 0; i < 4; i++)
831 | arr2[i + 4] = rl[i];
832 return (arr2);
833

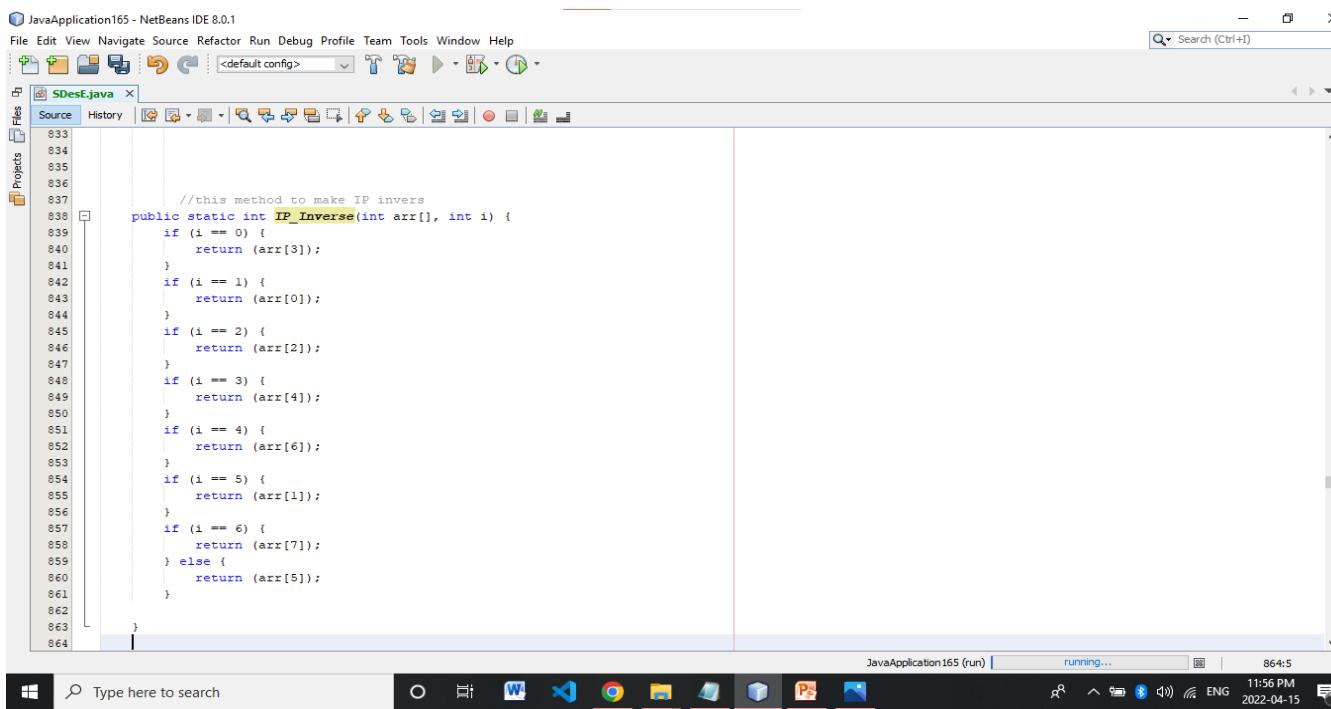
```



```
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689 // this method to convert to binary this use in sbox  
690 public static String binaary(int s) {  
691     if (s == 0) {  
692         return ("00");  
693     }  
694     if (s == 1) {  
695         return ("01");  
696     }  
697     if (s == 2) {  
698         return ("10");  
699     } else {  
700         return ("11");  
701     }  
702 }  
703  
704  
705  
706  
707  
708
```

Now we finished the function processes , let's continue with the encryption mode :

The output from the first iteration will enter the function again but this time with the key 2 and the same processes will happen again but this time the last X-Or will be between key2 and the first left part, then the output from the letter step will enter IP^{-1} which we will see its code :



```
833  
834  
835  
836  
837 //this method to make IP invers  
838 public static int IP_Inverse(int arr[], int i) {  
839     if (i == 0) {  
840         return (arr[3]);  
841     }  
842     if (i == 1) {  
843         return (arr[0]);  
844     }  
845     if (i == 2) {  
846         return (arr[2]);  
847     }  
848     if (i == 3) {  
849         return (arr[4]);  
850     }  
851     if (i == 4) {  
852         return (arr[6]);  
853     }  
854     if (i == 5) {  
855         return (arr[1]);  
856     }  
857     if (i == 6) {  
858         return (arr[7]);  
859     } else {  
860         return (arr[5]);  
861     }  
862 }  
863  
864
```

```

895
896
897     System.out.println("plain text itr2:");
898     // call f1 again
899     arr3 = fun(arr2, key2);
900     for (int i = 0; i < 4; i++) {
901         arr3l[i] = arr3[i];
902         arr3r[i] = arr3[i + 4];
903     }
904     for (int i = 0; i < 4; i++) {
905         arr3[i] = arr3r[i];
906         arr3[i + 4] = arr3l[i];
907     }
908     System.out.println("before ip invers:");
909     for (int i = 0; i < 8; i++) {
910         System.out.print(arr3[i]);
911     }
912     System.out.println();
913     // make IP invers to the final result
914     for (int i = 0; i < 8; i++) {
915         arr4[i] = IP_Inverse(arr3, i);
916     }
917     return arr4;
918 }
919
920
921
922
923
924
925
926

```

In this case the first encryption happened.

Now the second step is to do a decryption between the key2 and the output from the first step .

The decryption process is similar to the encryption process but all what we need is to replace the 2 keys , I mean we put key1 in the place of key2 , and the key2 in the place of key 1 as in the code bellow.

```

1118
1119 // the decryption method the same encryption but the order of key is different
1120 public static int[] dec(int arr[], int ke[]) {
1121
1122     int arr1[] = new int[8];
1123     int key[] = new int[8];
1124     int key2[] = new int[8];
1125     int arr2[] = new int[8];
1126     int arr3[] = new int[8];
1127     int arr3l[] = new int[4];
1128     int arr3r[] = new int[4];
1129     int arr4[] = new int[8];
1130     int keys[] = new int[16];
1131
1132     keys = gen(ke); // generate the keys
1133     for (int i = 0; i < 8; i++) {
1134         key[i] = keys[i];
1135         key2[i] = keys[i + 8];}
1136
1137     for (int i = 0; i < 8; i++) arr1[i] = IP(arr, i);
1138
1139     arr2 = fun(arr1, key2);
1140     System.out.println("the array after SW:");
1141     for (int i = 0; i < 8; i++) {
1142         System.out.print(arr2[i]);
1143     }
1144     System.out.println();
1145
1146     arr3 = fun(arr2, key);
1147     for (int i = 0; i < 4; i++) {
1148         arr3l[i] = arr3[i];
1149         arr3r[i] = arr3[i + 4];
1150     }

```

The screenshot shows the NetBeans IDE interface with a Java file named SDes.java open. The code implements the second step of the DES encryption process, which involves reversing the initial permutation (IP) of the 32-bit input block. The code includes comments explaining the steps:

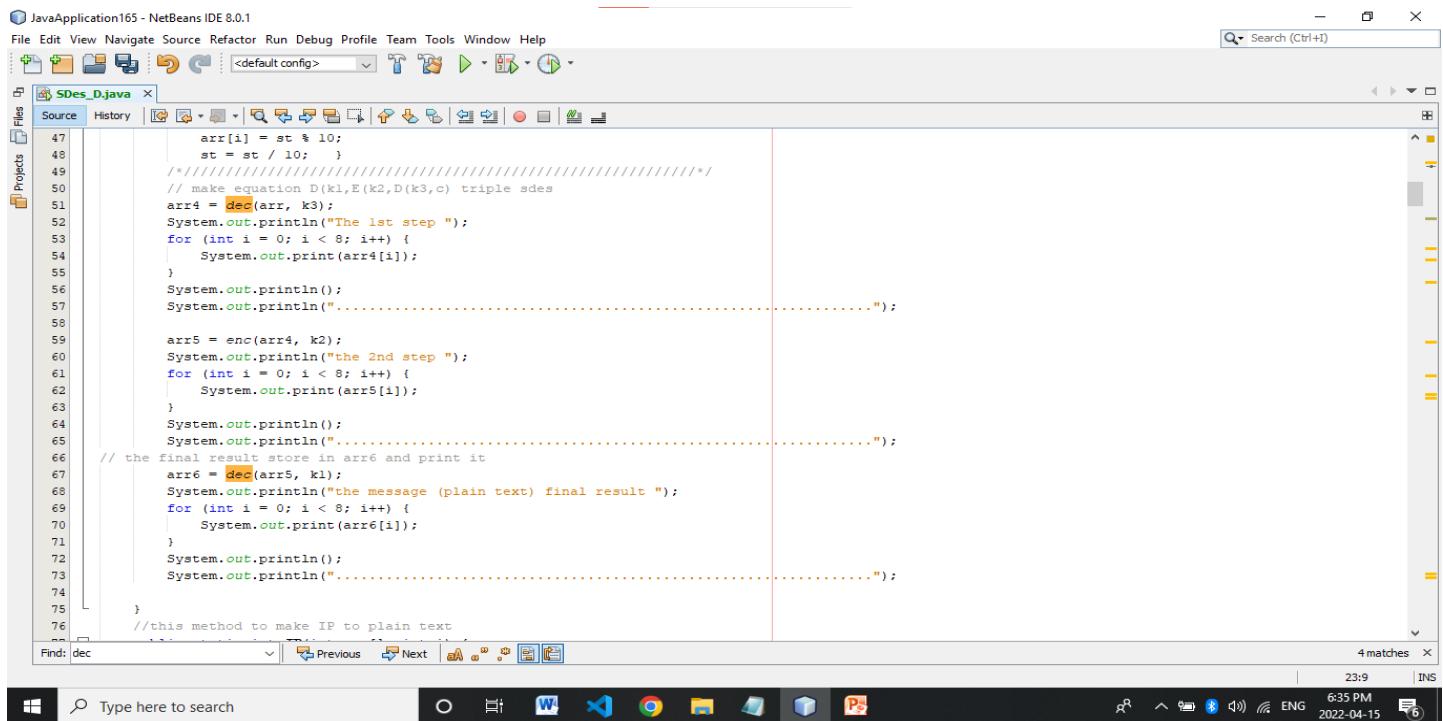
```
1151     for (int i = 0; i < 4; i++) {
1152         arr3[i] = arr3r[i];
1153         arr3[i + 4] = arr3l[i];
1154     }
1155     System.out.println("before ip invers:");
1156     for (int i = 0; i < 8; i++) {
1157         System.out.print(arr3[i]);
1158     }
1159     System.out.println();
1160
1161     for (int i = 0; i < 8; i++) {
1162         arr4[i] = IP_Inverse(arr3, i);
1163     }
1164
1165 }
1166 return arr4;
1167
1168 }
1169
1170 }
1171
1172
1173
1174 }
```

Now the last step is to do an encryption process between the output of the second step with key 3 , we talked about the encryption process before and how it works. Then the output from this step will be the cipher text that we need as bellow.

The screenshot shows the NetBeans IDE interface with the same Java file SDes.java open. The code now includes the final step of the DES process, which involves encrypting the result of the second step using key 3 (k3). The code prints the intermediate results and the final cipher text:

```
594
595
596
597     /*-----*/
598     // make equation E(k3,D(k2,E(k1,p) triple s des
599     arr4 = enc(arr, k1);
600     System.out.println("the step 1 result");
601     for (int i = 0; i < 8; i++) {
602         System.out.print(arr4[i]);
603     }
604     System.out.println();
605     System.out.println(".....");
606
607     arr5 = dec(arr4, k2);
608     System.out.println("the step 2 result ");
609     for (int i = 0; i < 8; i++) {
610         System.out.print(arr5[i]);
611     }
612     System.out.println();
613     System.out.println(".....");
614     // the final result store in arr6
615     arr6 = enc(arr5, k3);
616     System.out.println("the cipher text and the final result ");
617     for (int i = 0; i < 8; i++) {
618         System.out.print(arr6[i]);
619     }
620     System.out.println();
621     System.out.println(".....");
622
623 //this method to make ip to plain text
```

Now let's talk in a nutshell about the decryption of the triple S-DES because it's the same as the encryption the with this equation : $D(k3,E(k2,D(k1,p))$ instead of $E(k3,D(k2,E(k1,p))$ for the encryption with this code for the step :



The screenshot shows the NetBeans IDE interface with a Java file named 'SDes_D.java' open. The code implements the triple DES decryption process. It starts by dividing the input string into 10-character blocks. Then, it performs three steps: 1) Decryption using key3 (d_{des}), 2) Encryption using key2 (e_{des}), and 3) Decryption again using key3 (d_{des}). The results are printed to the console. The code includes comments explaining each step and the final result.

```

JavaApplication165 - NetBeans IDE 8.0.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
SDes_D.java <default config> Search (Ctrl+F)
Source History Files Projects
47     arr[i] = st % 10;
48     st = st / 10;
49     /*-----*/
50     // make equation D(k1,E(k2,D(k3,c)) triple sdes
51     arr4 = ddes(arr, k3);
52     System.out.println("The 1st step ");
53     for (int i = 0; i < 8; i++) {
54         System.out.print(arr4[i]);
55     }
56     System.out.println();
57     System.out.println(".....");
58
59     arr5 = enc(arr4, k2);
60     System.out.println("the 2nd step ");
61     for (int i = 0; i < 8; i++) {
62         System.out.print(arr5[i]);
63     }
64     System.out.println();
65     System.out.println(".....");
66     // the final result store in arr6 and print it
67     arr6 = ddes(arr5, k1);
68     System.out.println("the message (plain text) final result ");
69     for (int i = 0; i < 8; i++) {
70         System.out.print(arr6[i]);
71     }
72     System.out.println();
73     System.out.println(".....");
74
75 }
76 //this method to make IP to plain text

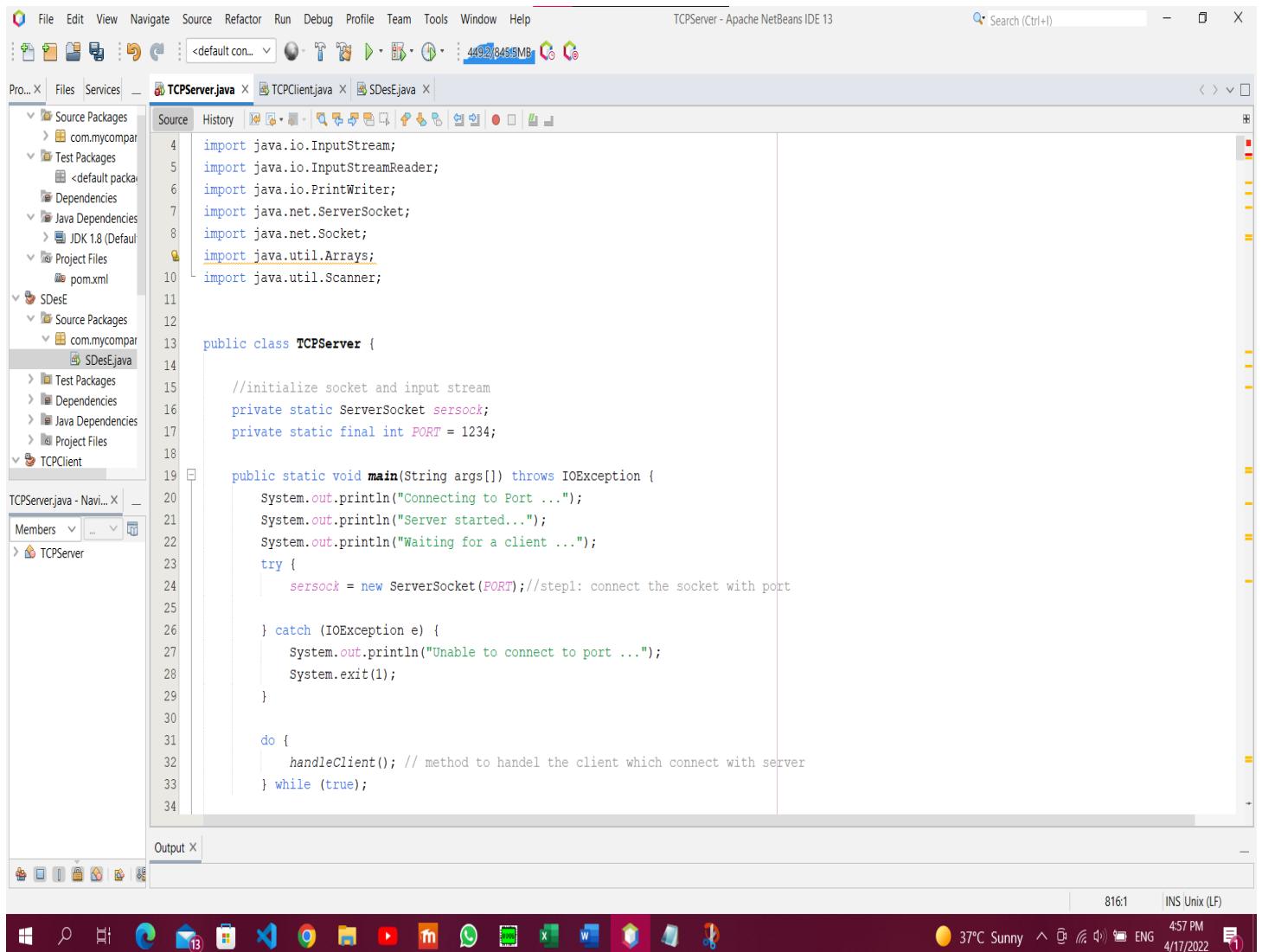
```

From the code above we can see that we have to start with the decryption process with the cipher text this time and then to do an encryption process using key2 and finally to do the decryption again with using the key 3 with exactly the same code and operations as those that we used in the encryption process .

The code below shows how to deal with a code to connect the client with the server .

First we can see the libraries those we used and needed.

The TCP Server code [Receiver Devise]: Firstly, the device will open its socket to specific port to connect with other device.

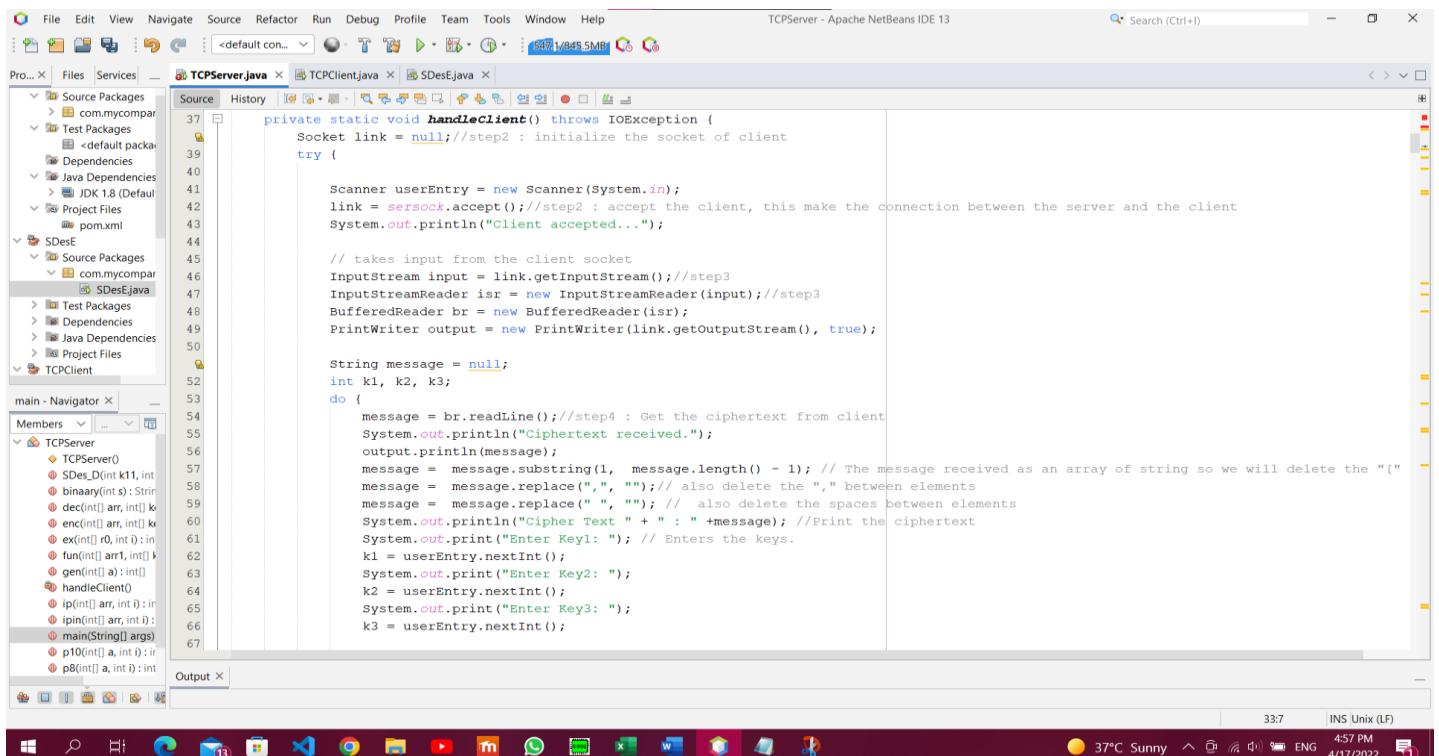


The screenshot shows the Apache NetBeans IDE 13 interface with the TCPServer.java file open in the editor. The code implements a TCP server that listens on port 1234 and handles client connections by calling a handleClient() method. The IDE's toolbars, project tree, and status bar are visible.

```
4 import java.io.InputStream;
5 import java.io.InputStreamReader;
6 import java.io.PrintWriter;
7 import java.net.ServerSocket;
8 import java.net.Socket;
9 import java.util.Arrays;
10 import java.util.Scanner;

11
12
13 public class TCPServer {
14
15     //initialize socket and input stream
16     private static ServerSocket sersock;
17     private static final int PORT = 1234;
18
19     public static void main(String args[]) throws IOException {
20         System.out.println("Connecting to Port ...");
21         System.out.println("Server started...");
22         System.out.println("Waiting for a client ...");
23         try {
24             sersock = new ServerSocket(PORT); //step1: connect the socket with port
25
26         } catch (IOException e) {
27             System.out.println("Unable to connect to port ...");
28             System.exit(1);
29         }
30
31         do {
32             handleClient(); // method to handle the client which connect with server
33         } while (true);
34     }
}
```

When the sender device connects with the receiver device, we call method handleClient()



The screenshot shows the Apache NetBeans IDE interface with the TCPClient.java file open in the editor. The code implements the handleClient() method, which initializes a socket, accepts a connection, reads ciphertext from the client, and prints it. It then prompts for three keys (k1, k2, k3) and calls the SDes_D() method to decrypt the message. The code also handles closing the connection and printing error messages for unhandled exceptions.

```

private static void handleClient() throws IOException {
    Socket link = null; //step2 : initialize the socket of client
    try {

        Scanner userEntry = new Scanner(System.in);
        link = sersock.accept(); //step2 : accept the client, this make the connection between the server and the client
        System.out.println("Client accepted...");

        // takes input from the client socket
        InputStream input = link.getInputStream(); //step3
        InputStreamReader isr = new InputStreamReader(input); //step3
        BufferedReader br = new BufferedReader(isr);
        PrintWriter output = new PrintWriter(link.getOutputStream(), true);

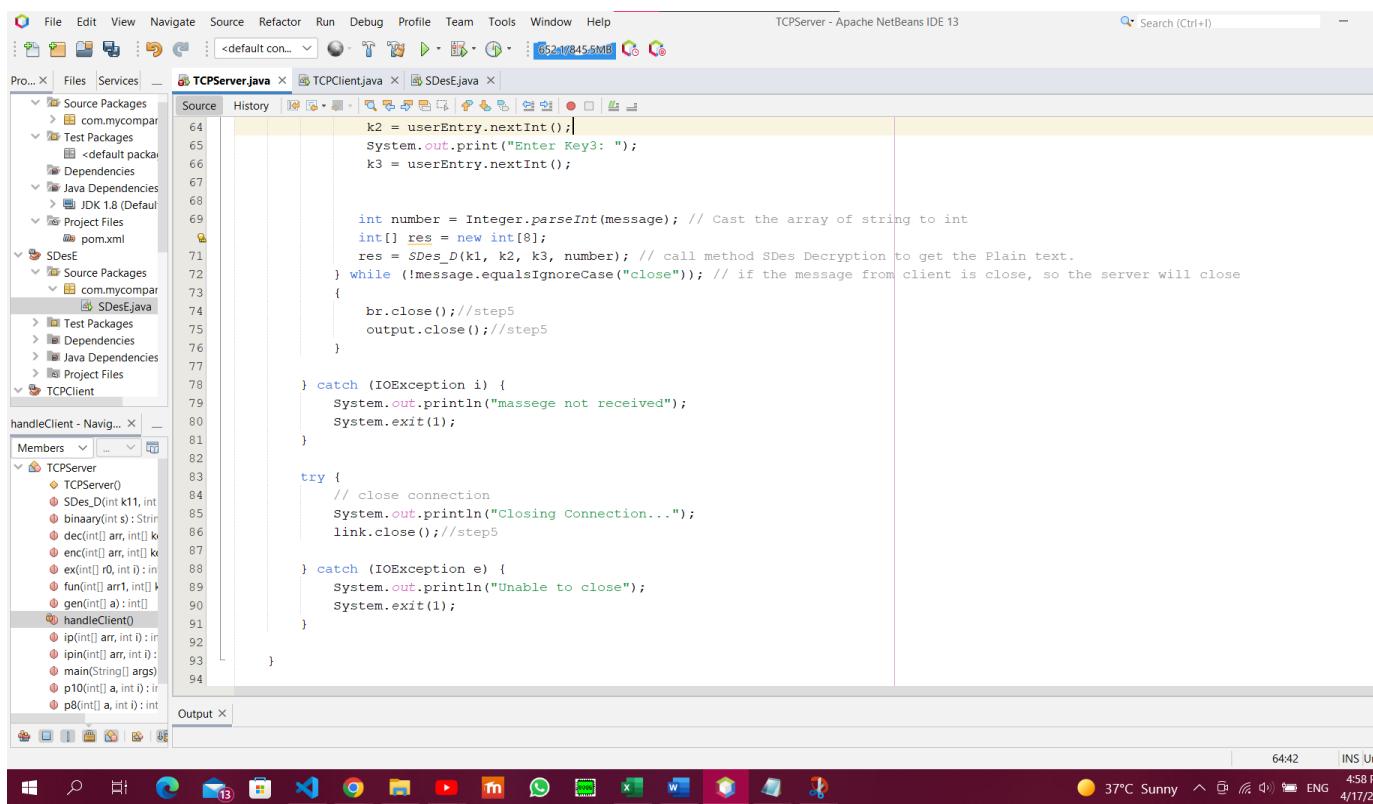
        String message = null;
        int k1, k2, k3;
        do {
            message = br.readLine(); //step4 : Get the ciphertext from client
            System.out.println("Ciphertext received.");
            output.println(message);
            message = message.substring(1, message.length() - 1); // The message received as an array of string so we will delete the "["
            message = message.replace(",", ""); // also delete the "," between elements
            message = message.replace(" ", ""); // also delete the spaces between elements
            System.out.println("Cipher Text " + ":" + message); //Print the ciphertext
            System.out.print("Enter Key1: "); // Enters the keys.
            k1 = userEntry.nextInt();
            System.out.print("Enter Key2: ");
            k2 = userEntry.nextInt();
            System.out.print("Enter Key3: ");
            k3 = userEntry.nextInt();
        } while (!message.equalsIgnoreCase("close")); // if the message from client is close, so the server will close
        {
            br.close(); //step5
            output.close(); //step5
        }

    } catch (IOException i) {
        System.out.println("massege not received");
        System.exit(1);
    }

    try {
        // close connection
        System.out.println("Closing Connection...");
        link.close(); //step5
    } catch (IOException e) {
        System.out.println("Unable to close");
        System.exit(1);
    }
}

```

Then the sender will send the cipher text. After that the receiver will enter the three keys.
 Then the cipher text will be decrypted by calling the decryption method to get the plain text.



The screenshot shows the Apache NetBeans IDE interface with the TCPClient.java file open in the editor. The code implements the handleClient() method, which initializes a socket, accepts a connection, reads ciphertext from the client, and prints it. It then prompts for three keys (k1, k2, k3) and calls the SDes_D() method to decrypt the message. The code also handles closing the connection and printing error messages for unhandled exceptions.

```

private static void handleClient() throws IOException {
    Socket link = null; //step2 : initialize the socket of client
    try {

        Scanner userEntry = new Scanner(System.in);
        link = sersock.accept(); //step2 : accept the client, this make the connection between the server and the client
        System.out.println("Client accepted...");

        // takes input from the client socket
        InputStream input = link.getInputStream(); //step3
        InputStreamReader isr = new InputStreamReader(input); //step3
        BufferedReader br = new BufferedReader(isr);
        PrintWriter output = new PrintWriter(link.getOutputStream(), true);

        String message = null;
        int k1, k2, k3;
        do {
            message = br.readLine(); //step4 : Get the ciphertext from client
            System.out.println("Ciphertext received.");
            output.println(message);
            message = message.substring(1, message.length() - 1); // The message received as an array of string so we will delete the "["
            message = message.replace(",", ""); // also delete the "," between elements
            message = message.replace(" ", ""); // also delete the spaces between elements
            System.out.println("Cipher Text " + ":" + message); //Print the ciphertext
            System.out.print("Enter Key1: "); // Enters the keys.
            k1 = userEntry.nextInt();
            System.out.print("Enter Key2: ");
            k2 = userEntry.nextInt();
            System.out.print("Enter Key3: ");
            k3 = userEntry.nextInt();
        } while (!message.equalsIgnoreCase("close")); // if the message from client is close, so the server will close
        {
            br.close(); //step5
            output.close(); //step5
        }

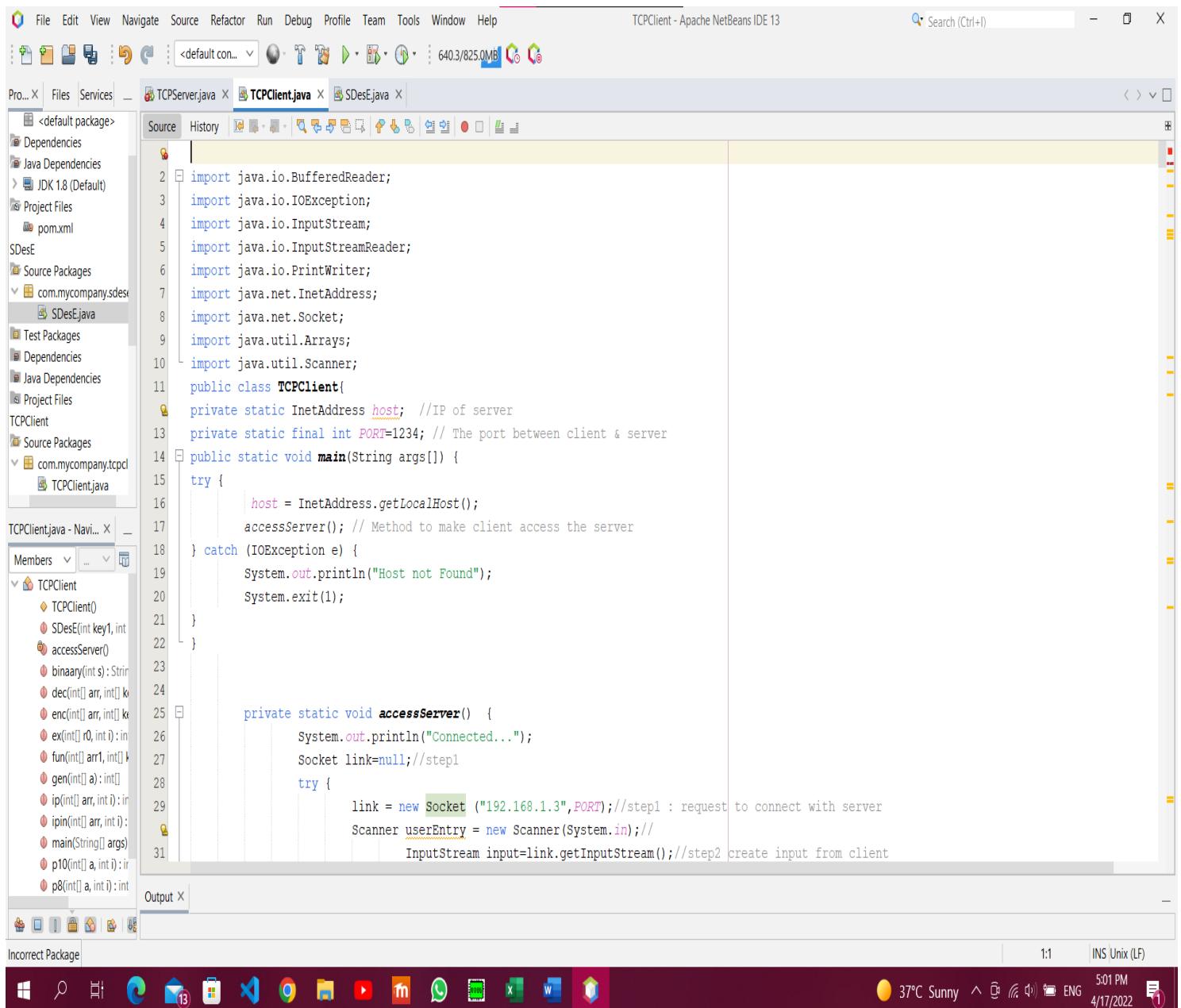
    } catch (IOException i) {
        System.out.println("massege not received");
        System.exit(1);
    }

    try {
        // close connection
        System.out.println("Closing Connection...");
        link.close(); //step5
    } catch (IOException e) {
        System.out.println("Unable to close");
        System.exit(1);
    }
}

```

The Sender Device [TCP Client code]:

Call method access server to send the cipher text. Basically, The sender device will connect with the receiver port using IP to make connection with. And initialize the parameters that we need.



The screenshot shows the Apache NetBeans IDE 13 interface with the TCPClient.java file open in the editor. The code implements a TCP client to connect to a server at 192.168.1.3 on port 1234. It uses BufferedReader and PrintWriter for communication and Scanner for user input. The code includes imports for java.io, java.net, and java.util. The main() method initializes the host and port, then attempts to connect to the server. If successful, it prints "Connected..." and reads input from the user.

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.InetAddress;
import java.net.Socket;
import java.util.Arrays;
import java.util.Scanner;
public class TCPClient{
    private static InetAddress host; //IP of server
    private static final int PORT=1234; // The port between client & server
    public static void main(String args[]) {
        try {
            host = InetAddress.getLocalHost();
            accessServer(); // Method to make client access the server
        } catch (IOException e) {
            System.out.println("Host not Found");
            System.exit(1);
        }
    }
    private static void accessServer() {
        System.out.println("Connected...");
        Socket link=null;//step1
        try {
            link = new Socket ("192.168.1.3",PORT);//step1 : request to connect with server
            Scanner userEntry = new Scanner(System.in);//step2
            InputStream input=link.getInputStream();//step2 create input from client
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

The sender will enter the three keys and the plain text. Then the plain text will be encrypted by calling encryption method to get the cipher text. Then the sender will send the cipher text to the server.

```

try {
    link = new Socket (host,PORT); //step1 : request to connect with server
    Scanner userEntry = new Scanner(System.in);//
    InputStream input=link.getInputStream(); //step2 create input from client
    InputStreamReader isr=new InputStreamReader(input);
    BufferedReader br=new BufferedReader(isr);
    PrintWriter output = new PrintWriter(link.getOutputStream()); // create output from user
    String ciphertext;
    int k1,k2,k3,pl;

    do {
        System.out.print("Enter Key1: "); // Now enters The keys
        k1 = userEntry.nextInt();
        System.out.print("Enter Key2: ");
        k2 = userEntry.nextInt();
        System.out.print("Enter Key3: ");
        k3 = userEntry.nextInt();
        System.out.print("Enter Plain Text: "); // Enter the plain text
        pl = userEntry.nextInt();
        ciphertext = (Arrays.toString(SDesE(k1,k2,k3,pl))); // Call method SDes Encryption
        output.println(ciphertext); // Send the ciphertext to the server As a string
        output.flush(); // Cleaning
        ciphertext=userEntry.next();
    } while (!ciphertext.equalsIgnoreCase("close")); // To close the connection between the server and client by type
    userEntry.close(); //step4 : close the system in
    br.close(); //step4
    output.close();
} catch (IOException e) {
    System.out.println("No response form server" );
}
}

```

If the sender enter close, so the socket will closed and that mean the end of connection between the devices.

```

try {
    System.out.print("Enter Key3: ");
    k3 = userEntry.nextInt();
    System.out.print("Enter Plain Text: "); // Enter the plain text
    pl = userEntry.nextInt();
    ciphertext = (Arrays.toString(SDesE(k1,k2,k3,pl))); // Call method SDes Encryption
    output.println(ciphertext); // Send the ciphertext to the server As a string
    output.flush(); // Cleaning
    ciphertext=userEntry.next();
} while (!ciphertext.equalsIgnoreCase("close")); // To close the connection between the server and client by type
userEntry.close(); //step4 : close the system in
br.close(); //step4
output.close();

} catch (IOException e) {
    System.out.println("No response form server" );
}

try {
    System.out.println("Close Connection...");
    link.close(); //step4 : close the socket of client
}
catch (IOException e) {
    System.out.println("Unable to connect");
    System.exit(1); // close the program if cant close the socket.
}

```

Verify correct work of our application:

Now let's check the results :

Network Programming in our Project

We have two devices:

1. Client (the sender).
2. Server (the receiver).

We connect them with network has TCP (Transmission Control Protocol).

- Hardware:
 - Ethernet.
 - Two devices.
- Software:

We used TCP code in java.

1. Run the server code from the cmd.

```
Command Prompt - java TCPServer
Microsoft Windows [Version 10.0.19044.1645]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Pascal>cd C:\Users\Pascal\Desktop\CryptographyProject

C:\Users\Pascal\Desktop\CryptographyProject>javac TCPServer.java

C:\Users\Pascal\Desktop\CryptographyProject>java TCPServer
```

2. The server will wait for a client to connect with.

```
C:\Users\Pascal\Desktop\CryptographyProject>java TCPServer
Connecting to Port ...
Server started...
Waiting for a client ...
```

3. Now run the client code from another cmd.

```
Command Prompt  
Microsoft Windows [Version 10.0.19044.1645]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\Pascal>cd C:\Users\Pascal\Desktop\CryptographyProject  
  
C:\Users\Pascal\Desktop\CryptographyProject>javac TCPClient.java  
  
C:\Users\Pascal\Desktop\CryptographyProject>java TCPClient  
Connected...
```

4. After run, the client will connect with server, then we want to enter the three keys and the plain text.

```
C:\Users\Pascal\Desktop\CryptographyProject>java TCPClient  
Connected...  
Enter Key1: 1111111111  
Enter Key2: 0000000000  
Enter Key3: 1010101010  
Enter Plain Text: 11110000
```

5. The plain text is encrypted. And now we have the cipher text, we want to send it to the server.

- Screen from client

```
The ciphertext and the final result  
01111111  
.....
```

- Screen from server after received the cipher text. And we want to enter the keys.

```
Client accepted...
Ciphertext received.
Cipher Text : 01111111
Enter Key1:
```

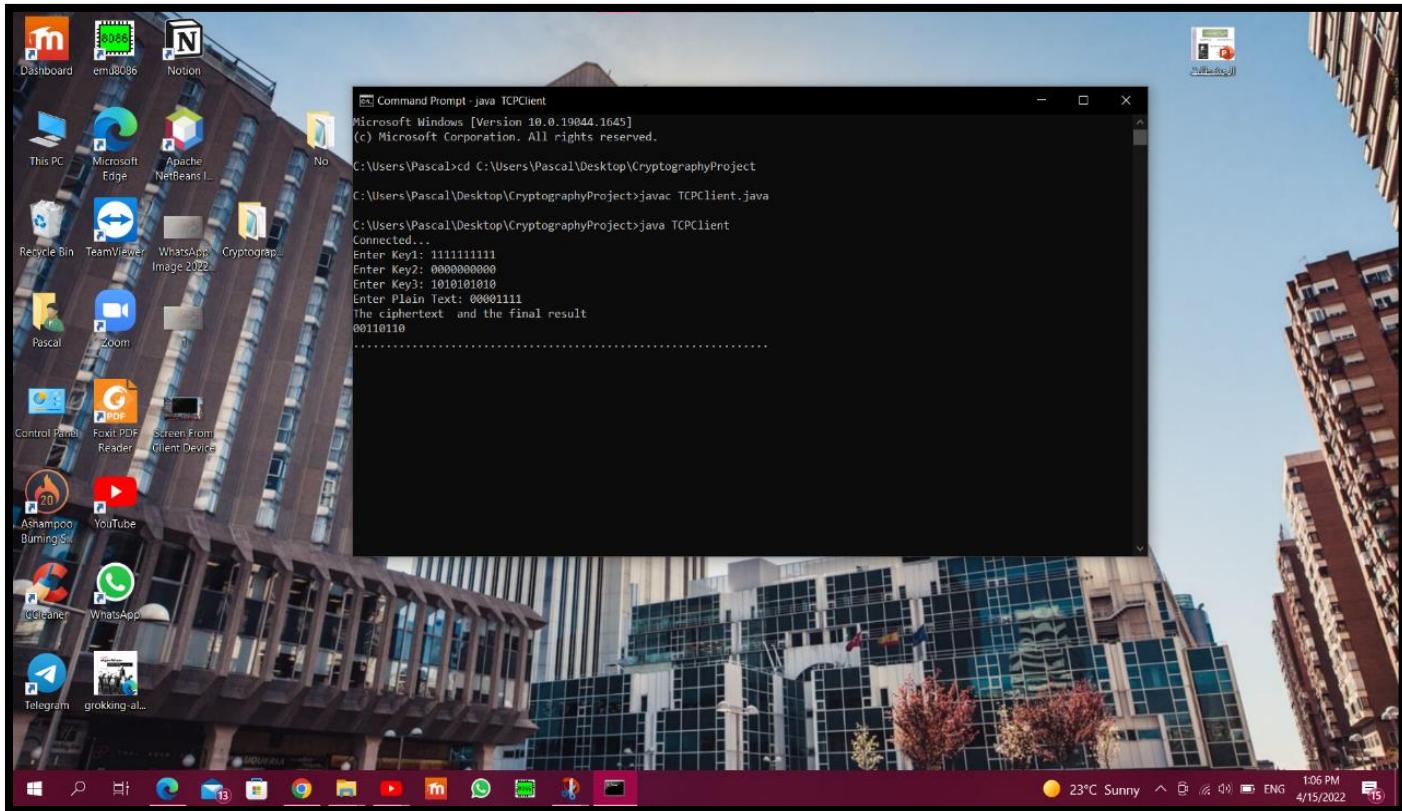
6. Entering the keys in server.

```
Cipher Text : 01111111
Enter Key1: 1111111111
Enter Key2: 0000000000
Enter Key3: 1010101010
```

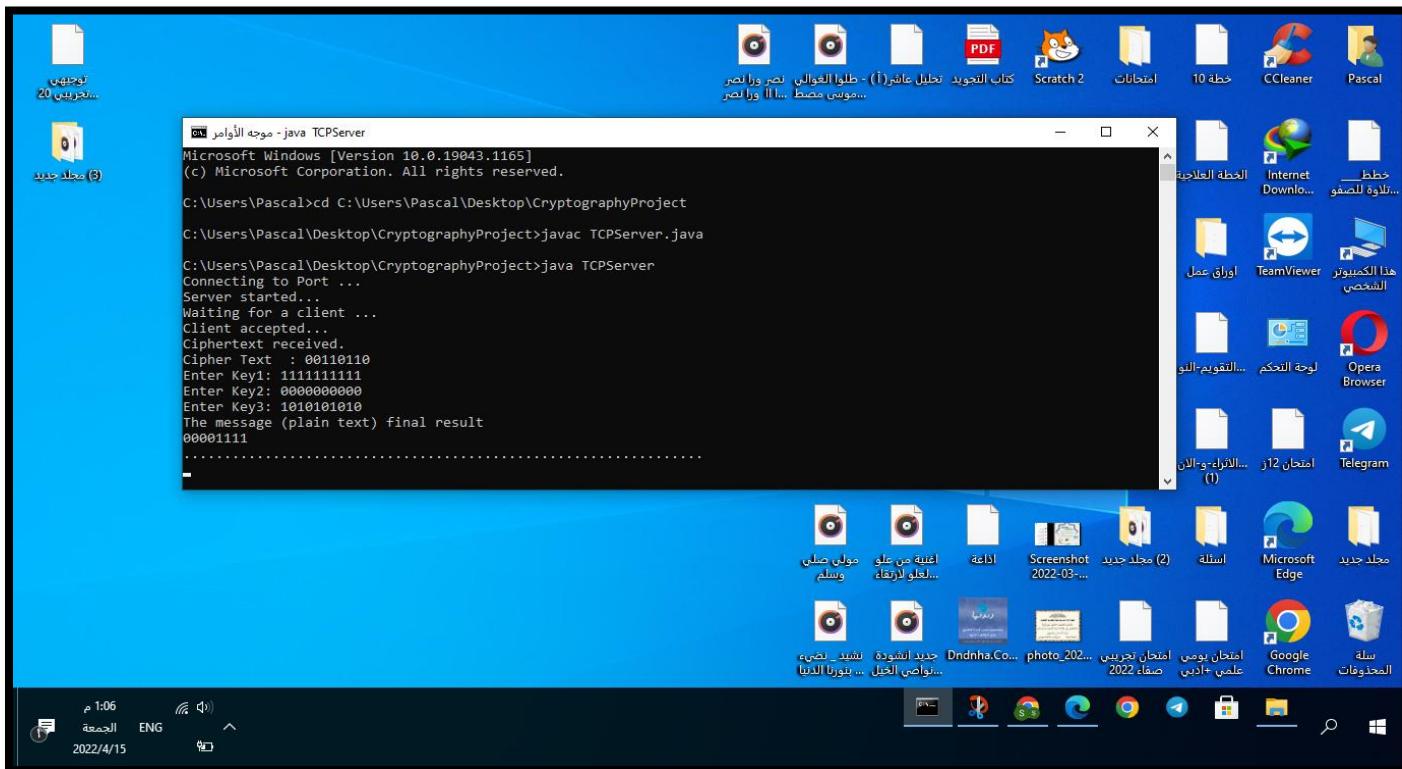
7. Now server have the encrypted message (cipher text) and the three keys, so the cipher text will be decrypted using triple S-Des to generate the plain text.

```
Cipher Text : 01111111
Enter Key1: 1111111111
Enter Key2: 0000000000
Enter Key3: 1010101010
The message (plain text) final result
11110000
.....
```

Screen from Sender Device



Screen from Receiver Device



Experiments results and discussion:

- As you can see from our codes , the language we used to write our codes is JAVA language since it's a high level language and it's easy to deal with especially because our code was kind of long.
- About the approach – triple S-DES – as we noticed is a complicated one but we are ok with it since it increases the security to transfer a message.
- Our message was just **bits** so we deal with bits only to encrypt and decrypt them not pics or normal letters, so it was sort of easy to deal with bits only . so because of that we didn't need the internet to help us with some ready codes or anything that will benefit us to write our triple S-DES code.

Conclusion:

- The code and the whole operation work in an consistent way.
- Using a high level language as JAVA made the deal with the code so easy and satisfying .
- Triple S-DES make the sender and receiver fell comfortable since it's more secure than the normal S-DES .
- Since the triple S-DES is a developed approach of the S-DES, writing the code and understanding the algorithm is easy and clear.

References:

- ✓ The slides of the cryptanalysis material from Dr. Anas Melhem .
- ✓ Using some codes that we learnt from network programming course .
- ✓ Java network programming 4th edition Elliotte rusty harold .