Network Security Homework 2

David Ayeke Jan 30. 2017

Network Security Homework 2

David Ayeke Jan 30. 2017

Problem 1.A: Encrypt "sendmoremoney with key stream "9 0 1 7 23 15 21 14 11 11 2 8 9"

Encrypted: beokjdmsxzpmh

Problem 1.B: Find a key such that above ciphertext decrypts to "cashnotneeded"

Key: [25, 4, 22, 3, 22, 15, 19, 5, 19, 21, 12, 8, 4]

Problem 1 Work

I created a simple program to perform the solve problem 1.

```
//encrypt takes a string and an array of numbers and returns the Vigenere ciphertext
function encrypt(str, key) {
  var encrypted = key.map(function(shift, i) {
    var normalized = str.charCodeAt(i) - 'a'.charCodeAt(0);
    var shifted = (normalized + shift) % 26;
    return String.fromCharCode('a'.charCodeAt(0) + shifted);
  });
  return encrypted.join("");
}

//findKey determines an array of numbers that will result in the ciphertext when encrypted by the
string
function findKey(str, ciphertext) {
  var key = str.split('').map(function(char, i) {
    var distance = ciphertext.charCodeAt(i) - char.charCodeAt(0)  ;
    if (distance < 0) {
      return 26 + distance;
    } else {
      return distance
    }
  });
  return key;
}

//decrypt returns the plaintext from the ciphertext and key.
function decrypt(str, key) {
  var encrypted = key.map(function(shift, i) {
    var normalized = str.charCodeAt(i) - 'a'.charCodeAt(0);
    var shifted = (normalized - shift);
    if (shifted < 0) {
      shifted = 26 + shifted;
    }
    return String.fromCharCode('a'.charCodeAt(0) + shifted);
  });
  return encrypted.join("");
}

var encrypted = encrypt('sendmoremoney', [9,0,1,7,23,15,21,14,11,11,2,8,9])
var test1 = decrypt(encrypted, [9,0,1,7,23,15,21,14,11,11,2,8,9]);
console.log(encrypted);
```

```
console.log("Test 1: Decrypt ciphertext", test1, test1 === 'sendmoremoney');
var newKey = findKey('cashnotneeded', encrypted);
var test2 = decrypt(encrypted, newKey);
console.log(newKey);
console.log("Test 2: Decrpyted ciphertext is now 'cashnotneeded'", test2, test2 === 'cashnotneeded');
var test3 = encrypt('cashnotneeded', newKey);
console.log("Test 3: 'cashnotneeded' has the same ciphertext as 'sendmoremoney'", test3, test3 ===
encrypted);
```

Part A was solved through the function call: encrypt('sendmoremoney', [9,0,1,7,23,15,21,14,11,11,2,8,9])

Part B was solved through the function call: findKey('cashnotneeded', encrypt('sendmoremoney', [9,0,1,7,23,15,21,14,11,11,2,8,9]));

HW 2.

plaintext = 0x0123456789abcdef key = 0x0123456789abcdef = 0000 0001 | 0000 0001 | 0100 0101 | 0110 0111 | 1000 1001 | 1010 1011 | 1100 1101 | 1110 1111

2.1 Derive K1

K1 = b02679b49a5 00001011000000100110011110011011010010011010010 1

2.2 Derive L0, R0

L0 = 1234567 R0 = 89abcdef

2.3 Expand R0 to get E[R0]

E[R0] = 7a15557a1555

2.4 Find A = E[R0]^K1

A = ['011100', '010001', '011100', '110010', '111000', '010101', '110011', '110000']

2.5 Find S-Box substitutions

S-Box substitution: [['0', '0', '0', '0'], ['1', '1', '0', '0'], ['0', '0', '1', '0'], ['0', '0', '0', '1'], ['0', '1', '1', '0'], ['1', '1', '0', '1'], ['0', '1', '0', '1'], ['0', '0', '0', '0']]

2.6 Concatenate S-box substitution for 32-bit result

B: c216d50

2.7 Apply the permutation to get P(B)

Permutation of B: 921c209c

2.8 Calculate R1 = P(B)^L0

R1: 5e1cec63

2.9 Write down ciphertext

ciphertext: f0aaf0aa5e1cec63

Problem 2 Work

I solved this problem by creating a javascript program.

```
// The program performs 1 round of DES and outputs several intermediate values
// run in command line with `node des.js`
// Scroll to the bottom for the full explaination of work.

// For programmer convinence, text is represented as an array of strings '0' or '1'
// The first element of this array is a space meening the first element of data occurs at index 1.
var plain_text = ["
","0","0","0","0","0","0","0","1","0","0","1","0","0","0","1","1","0","1","0","0","0","1","0","1","0","
1","1","0","0","1","1","1","1","0","0","0","1","0","0","1","1","0","1","0","1","0","1","1","1","1","0",
"0","1","1","0","1","1","1","1","0","1","1","1","1",]
var key = ["
","0","0","0","0","0","0","0","1","0","0","1","0","0","0","1","1","0","1","0","0","0","1","0","1","0","
1","1","0","0","1","1","1","1","0","0","0","1","0","0","1","1","0","1","0","1","0","1","1","1","1","0",
"0","1","1","0","1","1","1","1","0","1","1","1","1",]

// Calculates the inital perumutation.
function IP(t) {
  var permuted =
    t[58] +
    t[50] +
    t[42] +
    t[34] +
    t[26] +
    t[18] +
    t[10] +
    t[2] +
    t[60] +
    t[52] +
    t[44] +
    t[36] +
    t[28] +
    t[20] +
    t[12] +
    t[4] +
    t[62] +
    t[54] +
    t[46] +
    t[38] +
    t[30] +
    t[22] +
    t[14] +
    t[6] +
    t[64]+
    t[56] +
    t[48] +
    t[40] +
    t[32] +
    t[24] +
    t[16] +
    t[8] +
    t[57] +
    t[49] +
    t[41] +
    t[33] +
    t[25] +
    t[17] +
    t[9] +
    t[1] +
```

```
    t[59] +
    t[51] +
    t[43] +
    t[35] +
    t[27] +
    t[19] +
    t[11] +
    t[3] +
    t[61] +
    t[53] +
    t[45] +
    t[37] +
    t[29] +
    t[21] +
    t[13] +
    t[5] +
    t[63] +
    t[55] +
    t[47] +
    t[39] +
    t[31] +
    t[23] +
    t[15] +
    t[7];
  var toReturn = [].concat(" ", permuted.split(''));
  spacecheck(toReturn.slice(1,toReturn.length));
  return toReturn;
}

function get_first_56(key) {
  return [].concat(key.slice(1,8), key.slice(9,16), key.slice(17,24), key.slice(25,32),
key.slice(33,40), key.slice(41,48), key.slice(49,56), key.slice(57,64));
}

// Calculates the left half of PC1
function PC1_left(k) {
  var left_key =
    k[57] +
    k[49] +
    k[41] +
    k[33] +
    k[25] +
    k[17] +
    k[9] +
    k[1] +
    k[58] +
    k[50] +
    k[42] +
    k[34] +
    k[26] +
    k[18] +
    k[10] +
    k[2] +
    k[59] +
    k[51] +
    k[43] +
    k[35] +
    k[27] +
    k[19] +
    k[11] +
    k[3] +
```

```
    k[60] +
    k[52] +
    k[44] +
    k[36];
  var toReturn = [].concat([" "], left_key.split(""));
  spacecheck(toReturn.slice(1,toReturn.length));
  return toReturn;
}

// Calculates the left half of PC2
function PC1_right(k) {

  var right_key =
  k[63] +
  k[55] +
  k[47] +
  k[39] +
  k[31] +
  k[23] +
  k[15] +
  k[7] +
  k[62] +
  k[54] +
  k[46] +
  k[38] +
  k[30] +
  k[22] +
  k[14] +
  k[6] +
  k[61] +
  k[53] +
  k[45] +
  k[37] +
  k[29] +
  k[21] +
  k[13] +
  k[5] +
  k[28] +
  k[20] +
  k[12] +
  k[4];
  var toReturn = [].concat([" "], right_key.split(""));
  spacecheck(toReturn.slice(1,toReturn.length));
  return toReturn;
}

function arrayRotate(arr, count) {
  count -= arr.length * Math.floor(count / arr.length)
  arr.push.apply(arr, arr.splice(0, count))
  return arr
}

function rotate(k, ammount) {
  var arr = k.slice(1,k.length);
  var toReturn =  [].concat([" "], arrayRotate(arr, 1));
  spacecheck(toReturn.slice(1,toReturn.length));
  return toReturn;
}

// Calculates PC2
function PC2(left, right) {
```

```
  var k = [].concat([" "], left.slice(1, left.length), right.slice(1, right.length));
  var newKey =
    k[14] +
    k[17] +
    k[11] +
    k[24] +
    k[1] +
    k[5] +
    k[3] +
    k[28] +
    k[15] +
    k[6] +
    k[21] +
    k[10] +
    k[23] +
    k[19] +
    k[12] +
    k[4] +
    k[26] +
    k[8] +
    k[16] +
    k[7] +
    k[27] +
    k[20] +
    k[13] +
    k[2] +
    k[41] +
    k[52] +
    k[31] +
    k[37] +
    k[47] +
    k[55] +
    k[30] +
    k[40] +
    k[51] +
    k[45] +
    k[33] +
    k[48] +
    k[44] +
    k[49] +
    k[39] +
    k[56] +
    k[34] +
    k[53] +
    k[46] +
    k[42] +
    k[50] +
    k[36] +
    k[29] +
    k[32];

  var toReturn = [].concat(" ", newKey.split(''));
  spacecheck(toReturn.slice(1,toReturn.length));
  return toReturn;
}

function spacecheck(k) {
  k.forEach(function(char) {
    if (char !== "1" && char !=="0" ) {
      throw "Found Space: " + char;
    }
```

```
  });
}

// Returns the first 32 bits
function left(text){
  text = text.slice(1,text.length);
  var toReturn = [].concat(" ", text.slice(0,32));
  spacecheck(toReturn.slice(1,toReturn.length));
  return toReturn;
}

// Returns the last 32 bits
function right(text){
  text = text.slice(1,text.length);
  var toReturn = [].concat(" ", text.slice(32,64));
  spacecheck(toReturn.slice(1,toReturn.length));
  return toReturn;
}

// expands a 32 bit number into that of a 48 bit one.
function expand(text) {
  var expanded = [].concat(" ",
    text[32],
    text.slice(1,6),
    text.slice(4,10),
    text.slice(8, 14),
    text.slice(12,18),
    text.slice(16, 22),
    text.slice(20, 26),
    text.slice(24, 30),
    text.slice(28, 33),
    text[1]
  )
  spacecheck(expanded.slice(1,expanded.length));
  return expanded
}

function chunk(str, n) {
    var ret = [];
    var i;
    var len;

    for(i = 0, len = str.length; i < len; i += n) {
       ret.push(str.substr(i, n))
    }

    return ret
};

// Performes an xor operation.
function mix(text, key) {
  function xor(a,b) {
    if ((a == "1" && b == "0") || (a == "0" && b == "1")) {
      return "1";
    } else {
      return "0";
    }
  }
  var toReturn = [" "]
  for (var i = 1; i < text.length; i++ ) {
    toReturn.push(xor(text[i], key[i]));
```

```
  }
  spacecheck(toReturn.slice(1,toReturn.length));
  return toReturn;
}

var sb_1 = [14,4 ,13,1,2,15,11,8,3,10,6,12,5,9,0,7,
0,15,7,4,14,2,13,1,10,6,12,11,9,5,3,8,
4,1,14,8,13,6,2,11,15,12,9,7,3,10,5,0,
15,12,8,2,4,9,1,7,5,11,3,14,10,0,6,13]

var sb_2 = [15,1,8,14,6,11,3,4,9,7,2,13,12,0,5,10
,3,13,4,7,15,2,8,14,12,0,1,10,6,9,11,5
,0,14,7,11,10,4,13,1,5,8,12,6,9,3,2,15
,13,8,10,1,3,15,4,2,11,6,7,12,0,5,14,9]

var sb_3 = [10,0,9,14,6,3,15,5,1,13,12,7,11,4,2,8
,13,7,0,9,3,4,6,10,2,8,5,14,12,11,15,1
,13,6,4,9,8,15,3,0,11,1,2,12,5,10,14,7
,1,10,13,0,6,9,8,7,4,15,14,3,11,5,2,12]

var sb_4 = [7,13,14,3,0,6,9,10,1,2,8,5,11,12,4,15
,13,8,11,5,6,15,0,3,4,7,2,12,1,10,14,9
,10,6,9,0,12,11,7,13,15,1,3,14,5,2,8,4
,3,15,0,6,10,1,13,8,9,4,5,11,12,7,2,14]

var sb_5 = [2,12,4,1,7,10,11,6,8,5,3,15,13,0,14,9
,14,11,2,12,4,7,13,1,5,0,15,10,3,9,8,6
,4,2,1,11,10,13,7,8,15,9,12,5,6,3,0,14
,11,8,12,7,1,14,2,13,6,15,0,9,10,4,5,3]

var sb_6 = [12,1,10,15,9,2,6,8,0,13,3,4,14,7,5,11
,10,15,4,2,7,12,9,5,6,1,13,14,0,11,3,8
,9,14,15,5,2,8,12,3,7,0,4,10,1,13,11,6
,4,3,2,12,9,5,15,10,11,14,1,7,6,0,8,13]

var sb_7 = [4,11,2,14,15,0,8,13,3,12,9,7,5,10,6,1
,13,0,11,7,4,9,1,10,14,3,5,12,2,15,8,6
,1,4,11,13,12,3,7,14,10,15,6,8,0,5,9,2
,6,11,13,8,1,4,10,7,9,5,0,15,14,2,3,12]

var sb_8 = [13,2,8,4,6,15,11,1,10,9,3,14,5,0,12,7
,1,15,13,8,10,3,7,4,12,5,6,11,0,14,9,2
,7,11,4,1,9,12,14,2,0,6,10,13,15,3,5,8
,2,1,14,7,4,10,8,13,15,12,9,0,3,5,6,11
]

var sub_table = [sb_1, sb_2, sb_3, sb_4, sb_5, sb_6, sb_7, sb_8]

function nibble_to_binary(num) {
  var nbmap = {
    "0": ["0","0","0","0"] ,
    "1": ["0","0","0","1"],
    2: ["0","0","1","0"],
    3: ["0","0","1","1"],
    4: ["0","1","0","0"],
    5: ["0","1","0","1"],
    6: ["0","1","1","0"],
    7: ["0","1","1","1"],
    8: ["1","0","0","0"],
    9: ["1","0","0","1"],
    10: ["1","0","1","0"],
```

```
    11: ["1","0","1","1"],
    12: ["1","1","0","0"],
    13: ["1","1","0","1"],
    14: ["1","1","1","0"],
    15: ["1","1","1","1"],
  }
  return nbmap[num];
}

function sub_table_index(arr) {
  var rowmap = {"00": 0, "01": 1, "10": 2, "11": 3};
  var colmap = {
    "0000" : 0,
    "0001" : 1,
    "0010" : 2,
    "0011" : 3,
    "0100" : 4,
    "0101" : 5,
    "0110" : 6,
    "0111" : 7,
    "1000" : 8,
    "1001" : 9,
    "1010" : 10,
    "1011" : 11,
    "1100" : 12,
    "1101" : 13,
    "1110" : 14,
    "1111" : 15

  }
  var row = arr[0] + arr[5];
  var col = arr.slice(1,5);
  var toReturn = rowmap[row] * 16 + colmap[col];
  return toReturn
}

// Divide array into b
function substitute(text) {
  text = text.slice(1, text.length);
  var boxes = chunk(text.join(""), 6);
  var mapped = boxes.map(function(box, i) {
    var table = sub_table[i];
    var newValue = nibble_to_binary(table[sub_table_index(box)])
    return newValue;
  });
  return mapped;
}

// Permutation of a 32 bit number.
function P(text) {
  var table = [16,7,20,21,29,12,28,17,
    1,15,23,26,5,18,31,10,
    2,8,24,14,32,27,3,9,
    19,13,30,6,22,11,4,25]
  toReturn = text.map(function(value, i){
    if (i == 0) {
      return " "
    }
    var ret = text[table[i-1]];
    return ret;
  });
```

```
  spacecheck(toReturn.slice(1,toReturn.length));
  return toReturn;
}

function slice(text) {
  return text.slice(1, text.length);
}

/*
  K1 = PC2( PC1-Left(key) << 1, PC2-Right(key) << 1 )
  To solve for K1, I retrieved both left and right results of PC2.
  Each result was rotated left once.
  The results were combined and run through PC2.
*/
var left_key = rotate(PC1_left(key), 1);
var right_key = rotate(PC1_right(key), 1);
var sub_key1 = PC2(left_key, right_key);

// Answer for key 1
var ans1 = parseInt(sub_key1.slice(1, sub_key1.length).join(""), 2).toString(16);
console.log(ans1,sub_key1.slice(1, sub_key1.length).join(""));

/*
  L0 and L1 are the first 32 and last 32 bits of plaintext after it has been run
  through the initial permutation.
*/
var l0 = left(IP(plain_text));
var r0 = right(IP(plain_text));
var ans2_left = parseInt(l0.slice(1, sub_key1.length).join(""),2).toString(16);
var ans2_right = parseInt(r0.slice(1, sub_key1.length).join(""),2).toString(16);
console.log("l0:", ans2_left);
console.log("r0:", ans2_right);

/*
  E[R0] is caluclated with the expand function.
*/
var expanded = expand(r0);
var ans3 = parseInt(expanded.slice(1, expanded.length).join(""),2).toString(16);
console.log("Expanded: ", ans3, expanded.length);

/*
  A is the result of an xor operation with the expanded r0 and this round's key.
*/
var key_mixed = mix(expanded, sub_key1);
var ans4 = chunk(key_mixed.slice(1, key_mixed.length).join(''), 6);
console.log("A: ", ans4);

/*
  Substutition is performed with the substitution function
*/
var ans5 = substitute(key_mixed);
console.log("S-Box substitution: ", ans5);

var B = [].concat.apply([], ans5);
var ans6 = parseInt(B.join(""),2).toString(16);
console.log("B: ", ans6);

/*
 B is run through one last permutation before it becomes part of the ciphertext
*/
var PB = P([].concat([" "],B));
```

```
var ans7 = parseInt(PB.slice(1, PB.length).join(""),2).toString(16);
console.log("Permutation of B: ", ans7);

var R1 = mix(PB, l0);
var ans8 = parseInt(slice(R1).join(""),2).toString(16);
console.log("R1:", ans8);

/*
  Ciphertext is P(B) + L0
*/
var ans9 = ans2_right + ans8;
console.log("ciphertext:", ans9);
```