SERVER

Address: http://test.scytlbrasil.com/

Port: ***

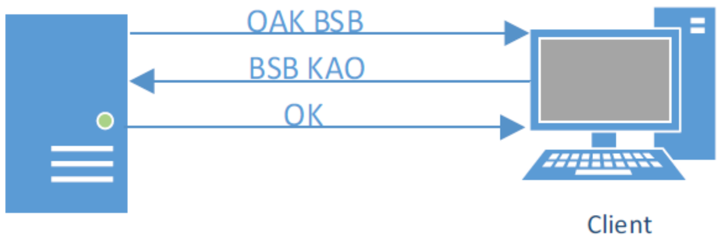

Table 1	
In	Out
0000	11110
0001	01001
0010	10100
0011	10101
0100	01010
0101	01011
0110	01110
0111	01111
1000	10010
1001	10011
1010	10110
1011	10111
1100	11010
1101	11011
1110	11100
1111	11101

Table 2	
Start pkt	11000110
End pkt	01101011
End Trans	00100001

Objective

1. Establish a connection with Scytl's server

2. Receive packets coded using "protocol X"

3. Decode the received packets thus obtaining the original message

4. Invert the original message

5. Re-encode the inverted message using the same "protocol X"

6. Send the resulting data back to the server

7. Receive confirmation message coded using "protocol X"

Sign up for the test

Access the website at the following URL: http://test.scytlbrasil.com

There you will be prompted for your name and email address. Once you have entered the requested data, you will be informed of the communication parameters you should use to connect with the server (host IP and port). You will also be able to view the packets sent and received between the server and your application.

Protocol X

Start packet	8 nibbles coded into 10 nibbles using Table 1					End Packet or End Transmission
0xC6						0x6B or 0x21

The protocol divides all data into 7-byte packets as following:

Example of encoding a single byte (2 nibbles):

'A' = 0x41 = 0100 0001 → 01010 01001

Notice that the encoding results in 10 bits, so the last 2 bits will be the first bits of the next byte.

To encode a message:

1. If the message length is not divisible by 4, pad to the right with spaces (0x20) as needed

2. Encode the padded message in ASCII

3. Take every 4 bytes of the ASCII-encoded message and transform into 5 bytes using Table 1

4. Add the StartPacket byte at the packet's beginning

5. Add the EndPacket byte at the packet's end if there are more bytes to transmit or the EndTransmission packet otherwise

To decode a message:

1. Divide the received bytes into packets using the StartPacket, EndPacket and EndTransmission packets

2. For each packet, take 5 bytes starting at the second byte and transform into 4 bytes using Table 1’s inverse

3. Concatenate the 4 bytes obtained from each packet into a single byte array

4. Decode the concatenated bytes as an ASCII string

5. Remove any trailing spaces (0x20) at the end of the string

Full example

Bytes received from server: 0xC6 0x57 0x54 0x95 0x5E 0x9E 0x6B 0xC6 0x55 0x17 0x55 0x52 0x9E 0x21

Decode message

Input: 0xC6 0x57 0x54 0x95 0x5E 0x9E 0x6B 0xC6 0x55 0x17 0x55 0x52 0x9E 0x21

Step 1 (divide the received bytes into packets using the StartPacket, EndPacket and EndTransmission packets):

0xC6 0x57 0x54 0x95 0x5E 0x9E 0x6B 0xC6 0x55 0x17 0x55 0x52 0x9E 0x21

0x57 0x54 0x95 0x5E 0x9E
0x55 0x17 0x55 0x52 0x9E

Step 2 (for each packet, take 5 bytes starting at the second byte and transform into 4 bytes using Table 1’s inverse):

Data bytes as 8-bit blocks

01010111 01010100 10010101 01011110 10011110
01010101 00010111 01010101 01010010 10011110

Bits regrouped into 5-bit blocks

01010 11101 01010 01001 01010 10111 10100 11110
01010 10100 01011 10101 01010 10100 10100 11110

5-bit blocks transformed into 4-bit blocks using Table 1’s inverse

0100 1111 0100 0001 0100 1011 0010 0000
0100 0010 0101 0011 0100 0010 0010 0000

Bits regrouped into 8-bit blocks

01001111 01000001 01001011 00100000

01000010 01010011 01000010 00100000

8-bit blocks as bytes

0x4F 0x41 0x4B 0x20

0x42 0x53 0x42 0x20

Step 3 (concatenate the 4 bytes obtained from each packet into a single byte array):

0x4F 0x41 0x4B 0x20 0x42 0x53 0x42 0x20

Step 4 (decode the concatenated bytes as an ASCII string):

"OAK BSB "

Step 5 (remove any trailing spaces (0x20) at the end of the string):

"OAK BSB" (decoding result)

Invert message

"OAK BSB"

"BSB KAO"

Encode inverted message

Input: "BSB KAO"

Step 1:

"BSB KAO "

Step 2:

0x42 0x53 0x42 0x20 0x4B 0x41 0x4F 0x20

Step 3:

0x42 0x53 0x42 0x20

0x4B 0x41 0x4F 0x20

01000010 01010011 01000010 00100000

01001011 01000001 01001111 00100000

```
0100 0010 0101 0011 0100 0010 0010 0000
0100 1011 0100 0001 0100 1111 0010 0000

01010 10100 01011 10101 01010 10100 10100 11110
01010 10111 01010 01001 01010 11101 10100 11110


01010101 00010111 01010101 01010010 10011110
01010101 11010100 10010101 01110110 10011110

0x55 0x17 0x55 0x52 0x9E
0x55 0xD4 0x95 0x76 0x9E
```

Steps 4/5:

0xC6 0x55 0x17 0x55 0x52 0x9E 0x6B

0xC6 0x55 0xD4 0x95 0x76 0x9E 0x21

0xC6 0x55 0x17 0x55 0x52 0x9E 0x6B 0xC6 0x55 0xD4 0x95 0x76 0x9E 0x21 (encoding result)

Response

If everything is done correctly, the server will return the message “OK” encoded in protocol X:
0xC6 0x57 0x55 0x7A 0x7A 0x9E 0x21

Otherwise, the server will return the message “ERR” encoded in protocol X:
0xC6 0x52 0xD7 0x45 0xD2 0x9E 0x21

Submit your solution

Once you successfully complete this test, send an email message to vagas.br@scytl.com with the same email address you used when submitting to the test, including the source code and instructions to compile the source code.

Need help?

You can view the packets sent and received between the server and your application on the test website.

If you need any assistance with understanding the test, please send a message to vagas.br@scytl.com

Good luck =)