RC4 is a broken algorithm and recommendation is to not use the same anymore if the data is to be kept highly secure.

If you still need a working implementation, you don't need to recreate the algorithm in your code. Java API javax.crypto can do it for you. Just generate a key and call the init method with mode set to encryption/decryption.

static String decryptRC4() throws NoSuchAlgorithmException, NoSuchPaddingException, InvalidKeyException, IllegalBlockSizeException, BadPaddingException, InvalidAlgorithmParameterException{

byte[] testDataBytes = "testString".getBytes();

KeyGenerator rc4KeyGenerator = KeyGenerator.getInstance("RC4");

SecretKey key = rc4KeyGenerator.generateKey();

// Create Cipher instance and initialize it to encrytion mode

Cipher cipher = Cipher.getInstance("RC4"); // Transformation of the algorithm

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

byte[] cipherBytes = cipher.doFinal(testDataBytes);

// Reinitialize the Cipher to decryption mode

cipher.init(Cipher.DECRYPT\_MODE,key, cipher.getParameters());

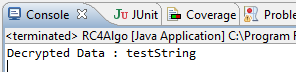
byte[] testDataBytesDecrypted = cipher.doFinal(cipherBytes);

System.out.println("Decrypted Data : "+new String(testDataBytesDecrypted));

return new String(testDataBytesDecrypted);

}

Output:

[](https://i.stack.imgur.com/H40qE.png)

If you need to send the encrypted data as part of a url then use Base64Encoding and then send.

e.g.

static String decryptRC4() throws NoSuchAlgorithmException, NoSuchPaddingException, InvalidKeyException, IllegalBlockSizeException, BadPaddingException, InvalidAlgorithmParameterException{

byte[] plainBytes = "testString".getBytes();

KeyGenerator rc4KeyGenerator = KeyGenerator.getInstance("RC4");

SecretKey key = rc4KeyGenerator.generateKey();

// Create Cipher instance and initialize it to encrytion mode

Cipher cipher = Cipher.getInstance("RC4"); // Transformation of the algorithm

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

byte[] cipherBytes = cipher.doFinal(plainBytes);

String encoded = encodeBase64(cipherBytes);

String decoded = decodeBase64(encoded);

// Reinitialize the Cipher to decryption mode

cipher.init(Cipher.DECRYPT\_MODE,key, cipher.getParameters());

byte[] plainBytesDecrypted = cipher.doFinal(Hex.decode(decoded));

System.out.println("Decrypted Data : "+new String(plainBytesDecrypted));

return new String(plainBytesDecrypted);

}

static String decodeBase64(String encodedData){

byte[] b = Base64.getDecoder().decode(encodedData);

String decodedData = DatatypeConverter.printHexBinary(b);

return decodedData;

}

static String encodeBase64(byte[] data){

byte[] b = Base64.getEncoder().encode(data);

String encodedData = new String(b);

/\*String encodedData = DatatypeConverter.printHexBinary(b);\*/

return encodedData;

}

**Tip:** Use Hex.decode as shown above to get bytes from the base64 decoded string or else you will get encoding issues. As much as possible do the conversions using Hex and convert to bytes array using bouncycastle methods.

Imports needed:

import java.io.IOException;

import java.security.InvalidAlgorithmParameterException;

import java.security.InvalidKeyException;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.util.Base64;

import javax.crypto.BadPaddingException;

import javax.crypto.Cipher;

import javax.crypto.IllegalBlockSizeException;

import javax.crypto.KeyGenerator;

import javax.crypto.NoSuchPaddingException;

import javax.crypto.SecretKey;

import javax.xml.bind.DatatypeConverter;

import org.apache.commons.codec.DecoderException;

import org.bouncycastle.util.encoders.Hex;

Also if you are generating a key from your own string you can use MD5Hashing for the same.

Please refer this for help on how to create a key using custom String: <https://stackoverflow.com/a/52463858/5912424>