DES Algorithm and Analysis of CCSCryptoUtil Project

Version 1.0
Date 03/11/10

Revision history

Doc No:	Ver sion no.	Change reference no.	Author	Published date	Sections changed	Description of changes
	1.0			03/11/10		Base Document

CONTENTS

Purpose of the document

This document describes the following

- Implementation of DES Algorithm
- Analysis of CCSCryptoUtilProject

Scope of the document

This document only touches the tip of a vast area of implementing cryptographic algorithms using the Java Crypto and Security API. It doesn't cover the following

- Implementation of other algorithms like AES etc.
- Details on Java Crypto and Security API.

Desired outcome

- Creating and implementing DES algorithm.

Targeted recipient

- Advanced developers.

Prerequisite

Recipients are expected to have knowledge on the following .

- Project creation.
- Adding libraries.
- Basic idea on Cryptography.

Required Software

- Apache common Codec library. (Can be found under "D:\Softwares\ALL JAR FILES\Apache\Common" in CCS7 machine).

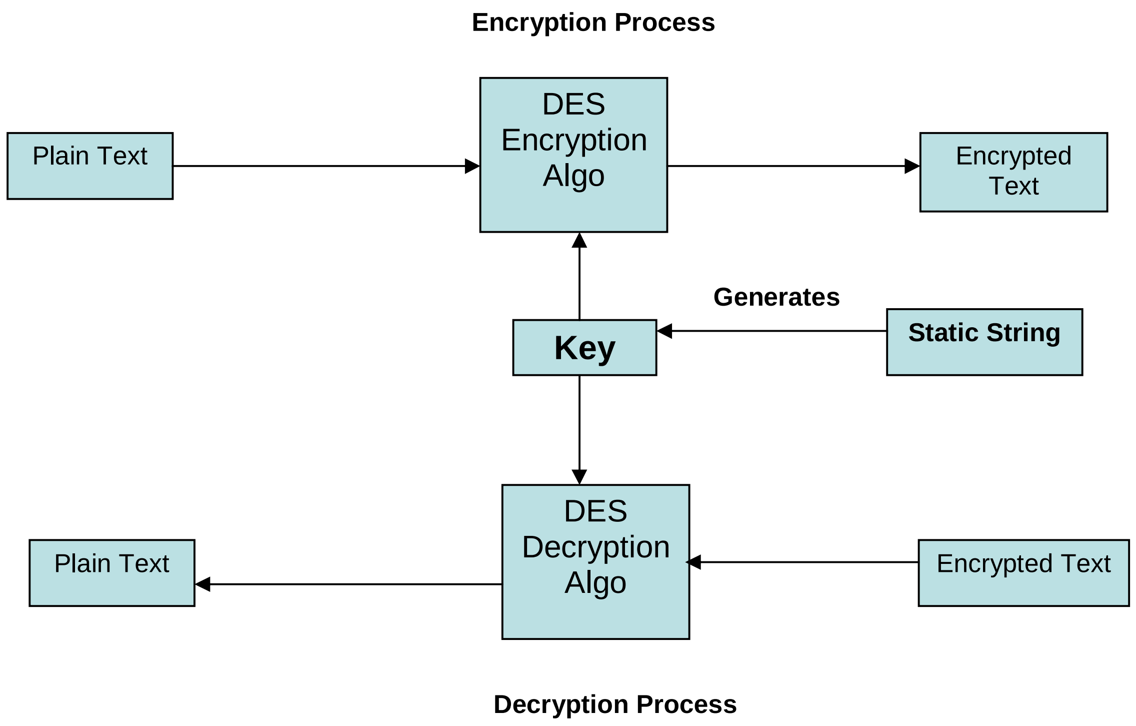
Cryptography Algorithm implemented in R+Plus

In order to prevent user's passwords from easily hacking by anyone having accessing to database, the passwords are stored in encrypted form.

Whenever a user supplies their passwords while logging in to the system the system decrypts the encrypted passwords stored in the database for verification.

The widely used DES algorithm is used for implementing the encryption and decryption algorithm. This algorithm is symmetric in the sense that the same key is used for both encryption and decryption.

A high level view of the process is illustrated below



As shown in the above figure the same key is used for both encryption and decryption algorithm. The key itself is generated based on a static string. Changing this string value will also change the generated key and will result in wrong decryption of the stored encrypted passwords.

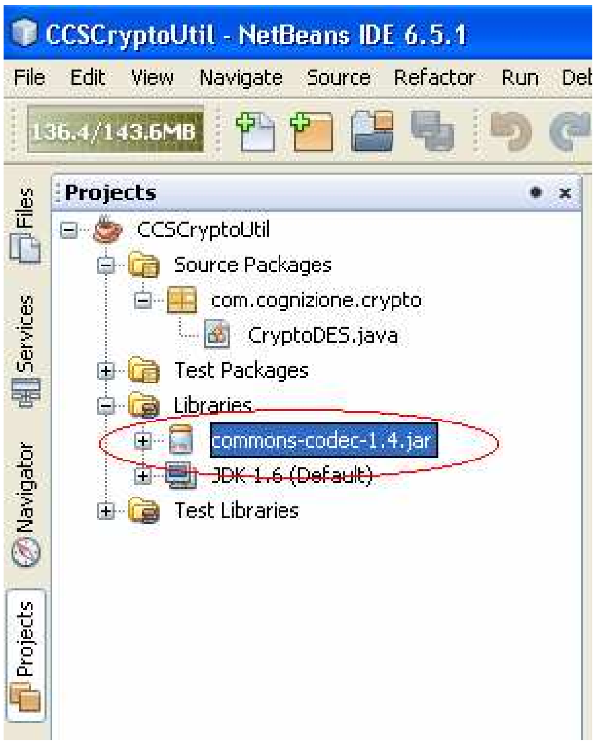
Java already provides an implementation of the DES algorithm so we don't need to implement it from scratch (As shown later).

The CCSCryptoUtil Project

In R+Plus the Cryptographic functions are provided by a jar file “CCSCryptoUtil.jar” file. This jar file is created from the “CCSCRyptoUtil” project.

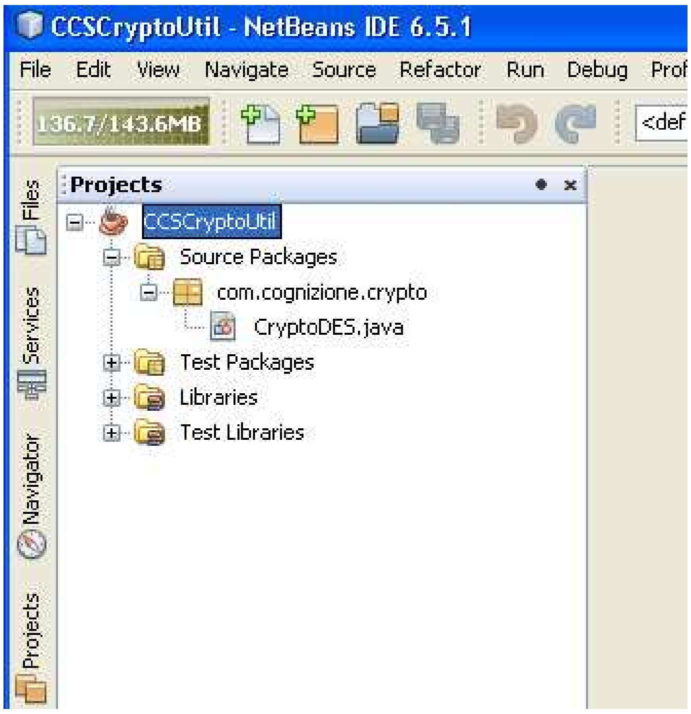
Project Dependency (CCSCryptoUtil)

The project depends on the Apache library – “common-codec-1.4”.



Analysis of the CCSCryptoUtil Project

The project contains one package “com.cognizione.crypto” and a single source file inside it – “CryptoDES”.



This class contains all the methods for encryption and decryption.

The variable to note in this class i.e. CryptoDES.java is the “*keyGenerator*” whose value is set to “abcdefghjklsnb”. This is the value which generates the key as shown in the first figure.

The length of this keyGenerator should be at least 8.

Changing the value of “keygenerator” will hence change the value of the key.

The complete source code is given below

Listing 1: CryptoDES.java

```
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.security.spec.InvalidKeySpecException;
import javax.crypto.Cipher;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.DESKeySpec;
import org.apache.commons.codec.binary.Base64;
```

```
/**
 *
 * Class which provides methods for encrypting and decrypting
 * strings using a DES encryption algorithm.
 * Strings can be encrypted and then are returned translated
 * into a Base64 Ascii String.
 *
 *  Date - 04/03/2010
 */
public class CryptoDES {

    private Cipher encryptCipher = null;
    private Cipher decryptCipher = null;
    private String keyGenerator = "abcdefghjklsnb"; //changing this value will create a new key
    private SecretKey key;

    /**
     * Construct a new object which can be utilized to encrypt
     * and decrypt strings using the specified key
     * with a DES encryption algorithm.
     *
     * @param key The secret key used in the crypto operations.
     *
     */

    public CryptoDES() throws InvalidKeyException, NoSuchAlgorithmException,
            InvalidKeySpecException, NoSuchPaddingException {

        DESKeySpec keySpec = new DESKeySpec(keyGenerator.getBytes());
        SecretKeyFactory keyFactory = SecretKeyFactory.getInstance("DES");
        key = keyFactory.generateSecret(keySpec);

        encryptCipher = Cipher.getInstance("DES");
        decryptCipher = Cipher.getInstance("DES");
        encryptCipher.init(Cipher.ENCRYPT_MODE, key);
        decryptCipher.init(Cipher.DECRYPT_MODE, key);

    }

    /**
     * Encrypt a string using DES encryption, and return the encrypted
     * string as a base64 encoded string.
     * @param unencryptedString The string to encrypt.
     * @return String The DES encrypted and base 64 encoded string.
     * @throws Exception If an error occurs.
     */
    public String encrypt(String unencryptedString) throws Exception {
        byte[] unencryptedByteArray = unencryptedString.getBytes("UTF8");
        byte[] encryptedBytes = encryptCipher.doFinal(unencryptedByteArray);
        byte[] encodedBytes = Base64.encodeBase64(encryptedBytes);

        return new String(encodedBytes);
    }

    /**
     * Decrypt a base64 encoded, DES encrypted string and return
     * the unencrypted string.
     * @param encryptedString The base64 encoded string to decrypt.
     * @return String The decrypted string.
     * @throws Exception If an error occurs.
     */

    public String decrypt(String encryptedString) throws Exception {
        byte[] decodedBytes = Base64.decodeBase64(encryptedString.getBytes());
        byte[] unencryptedByteArray = decryptCipher.doFinal(decodedBytes);

        return new String(unencryptedByteArray, "UTF8");
    }

}
```

Encrypting

To encrypt call the “encrypt” method passing a String i.e. the plain text. This will return a String in the encrypted form.

Decrypting

To decrypt call the “decrypt” method passing the encrypted String you want to decrypt. This will return a String in the plain text form.