JavaDigest
Java blog for avid java developers

RSA Encryption Example

Posted on **August 26, 2012**

Encryption is a one of the ways to achieve data security. Encryption is converting the data in the plain text into an unreadable text called the cipher text. *Decryption* is converting the cipher text back to plain text.

This encryption/decryption of data is part of *cryptography*. Encryption and decryption generally require the use of some secret information, referred to as a *key*, which is used in converting plain text to cipher text and vice versa.

Symmetric key cryptography refers to encryption methods in which both the sender and receiver share the same key.

Asymmetric key cryptography (also known as public key cryptography) refers to a system that requires two separate keys, one of which is secret and one of which is public. Although different, the two parts of the key pair are mathematically linked. One key encrypts the plain text, and the other decrypts the cipher text. Neither key can perform both functions. The public key, or the key used to encrypt information can be freely distributed.

RSA is one of the algorithm for public-key cryptography that is based on factoring large integers. RSA stands for Ron Rivest, Adi Shamir and Leonard Adleman, who first publicly described it.

Following example shows how to encrypt/decrypt information using RSA algorithm in Java.

The *KeyPairGenerator* class instance is used to generate the pair of public and private key for RSA algorithm and are saved into the files.

The *Cipher* class instance is used encrypt/decrypt information using the pair of keys generated above.

```
package in.javadigest.encryption;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.NoSuchAlgorithmException;
import java.security.PrivateKey;
import java.security.PublicKey;

import javax.crypto.Cipher;

/**
 * @author JavaDigest
 *
 */
public class EncryptionUtil {

  /**
   * String to hold name of the encryption algorithm.
   */
  public static final String ALGORITHM = "RSA";

  /**
   * String to hold the name of the private key file.
   */
  public static final String PRIVATE_KEY_FILE = "C:/keys/private.key";

  /**
   * String to hold name of the public key file.
   */
  public static final String PUBLIC_KEY_FILE = "C:/keys/public.key";

  /**
   * Generate key which contains a pair of private and public key using 1
   * bytes. Store the set of keys in Prvate.key and Public.key files.
   *
   * @throws NoSuchAlgorithmException
   * @throws IOException
   * @throws FileNotFoundException
   */
  public static void generateKey() {
    try {
      final KeyPairGenerator keyGen = KeyPairGenerator.getInstance(ALGOR
      keyGen.initialize(1024);
      final KeyPair key = keyGen.generateKeyPair();

      File privateKeyFile = new File(PRIVATE_KEY_FILE);
```

```
      File publicKeyFile = new File(PUBLIC_KEY_FILE);

      // Create files to store public and private key
      if (privateKeyFile.getParentFile() != null) {
        privateKeyFile.getParentFile().mkdirs();
      }
      privateKeyFile.createNewFile();

      if (publicKeyFile.getParentFile() != null) {
        publicKeyFile.getParentFile().mkdirs();
      }
      publicKeyFile.createNewFile();

      // Saving the Public key in a file
      ObjectOutputStream publicKeyOS = new ObjectOutputStream(
          new FileOutputStream(publicKeyFile));
      publicKeyOS.writeObject(key.getPublic());
      publicKeyOS.close();

      // Saving the Private key in a file
      ObjectOutputStream privateKeyOS = new ObjectOutputStream(
          new FileOutputStream(privateKeyFile));
      privateKeyOS.writeObject(key.getPrivate());
      privateKeyOS.close();
    } catch (Exception e) {
      e.printStackTrace();
    }

  }

  /**
   * The method checks if the pair of public and private key has been ge
   *
   * @return flag indicating if the pair of keys were generated.
   */
  public static boolean areKeysPresent() {

    File privateKey = new File(PRIVATE_KEY_FILE);
    File publicKey = new File(PUBLIC_KEY_FILE);

    if (privateKey.exists() && publicKey.exists()) {
      return true;
    }
    return false;
  }

  /**
   * Encrypt the plain text using public key.
   *
   * @param text
   *          : original plain text
   * @param key
   *          :The public key
   * @return Encrypted text
```

```
 * @throws java.lang.Exception
 */
public static byte[] encrypt(String text, PublicKey key) {
  byte[] cipherText = null;
  try {
    // get an RSA cipher object and print the provider
    final Cipher cipher = Cipher.getInstance(ALGORITHM);
    // encrypt the plain text using the public key
    cipher.init(Cipher.ENCRYPT_MODE, key);
    cipherText = cipher.doFinal(text.getBytes());
  } catch (Exception e) {
    e.printStackTrace();
  }
  return cipherText;
}

/**
 * Decrypt text using private key.
 *
 * @param text
 *          :encrypted text
 * @param key
 *          :The private key
 * @return plain text
 * @throws java.lang.Exception
 */
public static String decrypt(byte[] text, PrivateKey key) {
  byte[] dectyptedText = null;
  try {
    // get an RSA cipher object and print the provider
    final Cipher cipher = Cipher.getInstance(ALGORITHM);

    // decrypt the text using the private key
    cipher.init(Cipher.DECRYPT_MODE, key);
    dectyptedText = cipher.doFinal(text);

  } catch (Exception ex) {
    ex.printStackTrace();
  }

  return new String(dectyptedText);
}

/**
 * Test the EncryptionUtil
 */
public static void main(String[] args) {

  try {

    // Check if the pair of keys are present else generate those.
    if (!areKeysPresent()) {
      // Method generates a pair of keys using the RSA algorithm and s
      // in their respective files
```

```
            generateKey();
          }

          final String originalText = "Text to be encrypted ";
          ObjectInputStream inputStream = null;

          // Encrypt the string using the public key
          inputStream = new ObjectInputStream(new FileInputStream(PUBLIC_KEY_
          final PublicKey publicKey = (PublicKey) inputStream.readObject();
          final byte[] cipherText = encrypt(originalText, publicKey);

          // Decrypt the cipher text using the private key.
          inputStream = new ObjectInputStream(new FileInputStream(PRIVATE_KE
          final PrivateKey privateKey = (PrivateKey) inputStream.readObject(
          final String plainText = decrypt(cipherText, privateKey);

          // Printing the Original, Encrypted and Decrypted Text
          System.out.println("Original: " + originalText);
          System.out.println("Encrypted: " +cipherText.toString());
          System.out.println("Decrypted: " + plainText);

        } catch (Exception e) {
          e.printStackTrace();
        }
      }
    }
```

RELATED

Singleton Design Pattern
In "Design Pattern"

Using the sequence generator in Hibernate
In "Hibernate"

Generating Secure Random Numbers
In "Java"

This entry was posted in **Java** and tagged **Java**, **RSA Encryption**, **RSA Encryption Example** by **admin**. Bookmark the **permalink [https://javadigest.wordpress.com/2012/08/26/rsa-encryption-example/]** .

52 THOUGHTS ON “RSA ENCRYPTION EXAMPLE”

Hiren raval
on **March 12, 2013 at 11:12 am** said:

what are the public and private key file.???

admin
on **March 13, 2013 at 12:02 am** said:

Those files are just some means to store the public and the private keys respectively.
We should generating the the pair of keys only once for a pair of sender/receiver.
And use th
information

i am ge ed
to decr e
tell me

veerpal
on **February 7, 2015 at 6:11 pm** said:

```
package encrypt;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.NoSuchAlgorithmException;
import java.security.PrivateKey;
import java.security.PublicKey;

import javax.crypto.Cipher;

/**
* @author JavaDigest
*
*/
```

```
public class EncryptionUtil {

/**
* String to hold name of the encryption algorithm.
*/
public static final String ALGORITHM = “RSA”;

/**
* String to hold the name of the private key file.
*/
public static final String PRIVATE_KEY_FILE =
“C:/keys/private.key”;

/**
* String to hold name of the public key file.
*/
public static final String PUBLIC_KEY_FILE =
“C:/keys/public.key”;

/**
* Generate key which contains a pair of private
and public key using 1024
* bytes. Store the set of keys in Prvate.key and
Public.key files.
*
* @throws NoSuchAlgorithmException
* @throws IOException
* @throws FileNotFoundException
*/
public static void generateKey() {
try {
final KeyPairGenerator keyGen =
KeyPairGenerator.getInstance(ALGORITHM);
keyGen.initialize(1024);
final KeyPair key = keyGen.generateKeyPair();

File privateKeyFile = new
File(PRIVATE_KEY_FILE);
File publicKeyFile = new
File(PUBLIC_KEY_FILE);

// Create files to store public and private key
if (privateKeyFile.getParentFile() != null) {
privateKeyFile.getParentFile().mkdirs();
}
privateKeyFile.createNewFile();
```

```
if (publicKeyFile.getParentFile() != null) {
publicKeyFile.getParentFile().mkdirs();
}
publicKeyFile.createNewFile();

// Saving the Public key in a file
ObjectOutputStream publicKeyOS = new
ObjectOutputStream(
new FileOutputStream(publicKeyFile));
publicKeyOS.writeObject(key.getPublic());
publicKeyOS.close();

// Saving the Private key in a file
ObjectOutputStream privateKeyOS = new
ObjectOutputStream(
new FileOutputStream(privateKeyFile));
privateKeyOS.writeObject(key.getPrivate());
privateKeyOS.close();
} catch (Exception e) {
e.printStackTrace();
}

}

/**
* The method checks if the pair of public and
private key has been generated.
*
* @return flag indicating if the pair of keys were
generated.
*/
public static boolean areKeysPresent() {

File privateKey = new File(PRIVATE_KEY_FILE);
File publicKey = new File(PUBLIC_KEY_FILE);

if (privateKey.exists() && publicKey.exists()) {
return true;
}
return false;
}

/**
* Encrypt the plain text using public key.
*
* @param text
```

```
* : original plain text
* @param key
* :The public key
* @return Encrypted text
* @throws java.lang.Exception
*/
public static byte[] encrypt(String text, PublicKey
key) {
byte[] cipherText = null;
try {
// get an RSA cipher object and print the provider
final Cipher cipher =
Cipher.getInstance(ALGORITHM);
// encrypt the plain text using the public key
cipher.init(Cipher.ENCRYPT_MODE, key);
cipherText = cipher.doFinal(text.getBytes());
} catch (Exception e) {
e.printStackTrace();
}
return cipherText;
}

/**
* Decrypt text using private key.
*
* @param text
* :encrypted text
* @param key
* :The private key
* @return plain text
* @throws java.lang.Exception
*/
public static String decrypt(byte[] text, PrivateKey
key) {
byte[] dectyptedText = null;
try {
// get an RSA cipher object and print the provider
final Cipher cipher =
Cipher.getInstance(ALGORITHM);

// decrypt the text using the private key
cipher.init(Cipher.DECRYPT_MODE, key);
dectyptedText = cipher.doFinal(text);

} catch (Exception ex) {
ex.printStackTrace();
```

```
}

return new String(dectyptedText);
}

/**
* Test the EncryptionUtil
*/
public static void main(String[] args) {

try {

// Check if the pair of keys are present else
generate those.
if (!areKeysPresent()) {
// Method generates a pair of keys using the RSA
algorithm and stores it
// in their respective files
generateKey();
}

final String originalText = “veerpal”;
ObjectInputStream inputStream = null;

// Encrypt the string using the public key
inputStream = new ObjectInputStream(new
FileInputStream(PUBLIC_KEY_FILE));
final PublicKey publicKey = (PublicKey)
inputStream.readObject();
final byte[] cipherText = encrypt(originalText,
publicKey);

// Decrypt the cipher text using the private key.
inputStream = new ObjectInputStream(new
FileInputStream(PRIVATE_KEY_FILE));
final PrivateKey privateKey = (PrivateKey)
inputStream.readObject();
final String plainText = decrypt(cipherText,
privateKey);

// Printing the Original, Encrypted and Decrypted
Text
System.out.println(“Original: ” + originalText);
System.out.println(“Encrypted: ”
+cipherText.toString());
System.out.println(“Decrypted: ” + plainText);
```

```
} catch (Exception e) {
e.printStackTrace();
}
}
}
```

Anup
on **May 27, 2013 at 12:07 am** said:

Thank you for a concise and complete example discussing from key generation to encrypting and decrypting content. Much Appreciated.

Question: How would the encryption look like, if padding needs to be added to the encryption (to make it strong, as per RSA recommendations i read).

> **admin**
> on **May 28, 2013 at 9:40 pm** said:
>
> The Cipher.getInstance() method takes transformation string as its parameter.
> The transformation string can take two formats:
>
> A transformation string is of the form:
> “algorithm/mode/padding” or
> “algorithm”
>
> Currently i have used “RSA” (algorithm) as the transformation string.
> You can you “RSA/ECB/PKCS1PADDING” instead in the encrypt and decrypt method (need not change the generateKey method).
>
> You will need to change the code to;
>
> final Cipher cipher =
> Cipher.getInstance(“RSA/ECB/PKCS1PADDING”);

Anup
on **May 28, 2013 at 9:57 pm** said:

great.. got it..

Thanks for the response.

nick
on **July 7, 2013 at 10:11 pm** said:

this code works well but i am having challenges implementing in in socket programming how do i send the stream to the server. if you need to see the code i can provide it

admin
on **July 8, 2013 at 11:35 pm** said:

Are you trying to send key to the server ?
I believe you can use ObjectOutputStream to send key to the server.
Share the code if it is not working

nick
on **July 15, 2013 at 8:49 pm** said:

Below are the 3 classes Cryptography,
MyClientRsa and MyServerRsa
and when i type some text in the client the server
receives the cipher text but it cannot convert it
back to the correct plain text.

```
//this is the encryption class
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
```

```
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.NoSuchAlgorithmException;
import java.security.PrivateKey;
import java.security.PublicKey;

import javax.crypto.Cipher;

/**
* @author JavaDigest
*
*/
public class Cryptography{

/**
* String to hold name of the encryption algorithm.
*/
public static final String ALGORITHM = “RSA”;

/**
* String to hold the name of the private key file.
*/
public static final String PRIVATE_KEY_FILE =
“C:/keys/private.key”;

/**
* String to hold name of the public key file.
*/
public static final String PUBLIC_KEY_FILE =
“C:/keys/public.key”;

/**
* Generate key which contains a pair of private
and public key using 1024
* bytes. Store the set of keys in Prvate.key and
Public.key files.
*
* @throws NoSuchAlgorithmException
* @throws IOException
* @throws FileNotFoundException
*/
public static void generateKey() {
try {
```

```
final KeyPairGenerator keyGen =
KeyPairGenerator.getInstance(ALGORITHM);
keyGen.initialize(1024);
final KeyPair key = keyGen.generateKeyPair();

File privateKeyFile = new
File(PRIVATE_KEY_FILE);
File publicKeyFile = new
File(PUBLIC_KEY_FILE);

// Create files to store public and private key
if (privateKeyFile.getParentFile() != null) {
privateKeyFile.getParentFile().mkdirs();
}
privateKeyFile.createNewFile();

if (publicKeyFile.getParentFile() != null) {
publicKeyFile.getParentFile().mkdirs();
}
publicKeyFile.createNewFile();

// Saving the Public key in a file
ObjectOutputStream publicKeyOS = new
ObjectOutputStream(
new FileOutputStream(publicKeyFile));
publicKeyOS.writeObject(key.getPublic());
publicKeyOS.close();

// Saving the Private key in a file
ObjectOutputStream privateKeyOS = new
ObjectOutputStream(
new FileOutputStream(privateKeyFile));
privateKeyOS.writeObject(key.getPrivate());
privateKeyOS.close();
} catch (Exception e) {
e.printStackTrace();
}

}

/**
* The method checks if the pair of public and
private key has been generated.
*
* @return flag indicating if the pair of keys were
generated.
```

```
*/
public static boolean areKeysPresent() {

File privateKey = new File(PRIVATE_KEY_FILE);
File publicKey = new File(PUBLIC_KEY_FILE);

if (privateKey.exists() && publicKey.exists()) {
return true;
}
return false;
}

/**
* Encrypt the plain text using public key.
*
* @param text
* : original plain text
* @param key
* :The public key
* @return Encrypted text
* @throws java.lang.Exception
*/
public static byte[] encrypt(String text, PublicKey
key) {
byte[] cipherText = null;
try {
// get an RSA cipher object and print the provider
final Cipher cipher =
Cipher.getInstance(ALGORITHM);
// encrypt the plain text using the public key
cipher.init(Cipher.ENCRYPT_MODE, key);
cipherText = cipher.doFinal(text.getBytes());
} catch (Exception e) {
e.printStackTrace();
}
return cipherText;
}

/**
* Decrypt text using private key.
*
* @param text
* :encrypted text
* @param key
* :The private key
* @return plain text
```

```
* @throws java.lang.Exception
*/
public static String decrypt(byte[] text, PrivateKey
key) {
byte[] dectyptedText = null;
try {
// get an RSA cipher object and print the provider
final Cipher cipher =
Cipher.getInstance(ALGORITHM);

// decrypt the text using the private key
cipher.init(Cipher.DECRYPT_MODE, key);
dectyptedText = cipher.doFinal(text);

} catch (Exception ex) {
ex.printStackTrace();
}

return new String(dectyptedText);
}
}

// Server

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.NoSuchAlgorithmException;
import java.security.PrivateKey;
import java.security.PublicKey;

import javax.crypto.Cipher;

/**
* @author JavaDigest
*
*/
public class Cryptography{

/**
```

```
* String to hold name of the encryption algorithm.
*/
public static final String ALGORITHM = “RSA”;

/**
* String to hold the name of the private key file.
*/
public static final String PRIVATE_KEY_FILE =
“C:/keys/private.key”;

/**
* String to hold name of the public key file.
*/
public static final String PUBLIC_KEY_FILE =
“C:/keys/public.key”;

/**
* Generate key which contains a pair of private
and public key using 1024
* bytes. Store the set of keys in Prvate.key and
Public.key files.
*
* @throws NoSuchAlgorithmException
* @throws IOException
* @throws FileNotFoundException
*/
public static void generateKey() {
try {
final KeyPairGenerator keyGen =
KeyPairGenerator.getInstance(ALGORITHM);
keyGen.initialize(1024);
final KeyPair key = keyGen.generateKeyPair();

File privateKeyFile = new
File(PRIVATE_KEY_FILE);
File publicKeyFile = new
File(PUBLIC_KEY_FILE);

// Create files to store public and private key
if (privateKeyFile.getParentFile() != null) {
privateKeyFile.getParentFile().mkdirs();
}
privateKeyFile.createNewFile();

if (publicKeyFile.getParentFile() != null) {
publicKeyFile.getParentFile().mkdirs();
```

```
}
publicKeyFile.createNewFile();

// Saving the Public key in a file
ObjectOutputStream publicKeyOS = new
ObjectOutputStream(
new FileOutputStream(publicKeyFile));
publicKeyOS.writeObject(key.getPublic());
publicKeyOS.close();

// Saving the Private key in a file
ObjectOutputStream privateKeyOS = new
ObjectOutputStream(
new FileOutputStream(privateKeyFile));
privateKeyOS.writeObject(key.getPrivate());
privateKeyOS.close();
} catch (Exception e) {
e.printStackTrace();
}

}

/**
* The method checks if the pair of public and
private key has been generated.
*
* @return flag indicating if the pair of keys were
generated.
*/
public static boolean areKeysPresent() {

File privateKey = new File(PRIVATE_KEY_FILE);
File publicKey = new File(PUBLIC_KEY_FILE);

if (privateKey.exists() && publicKey.exists()) {
return true;
}
return false;
}

/**
* Encrypt the plain text using public key.
*
* @param text
* : original plain text
* @param key
```

```
* :The public key
* @return Encrypted text
* @throws java.lang.Exception
*/
public static byte[] encrypt(String text, PublicKey
key) {
byte[] cipherText = null;
try {
// get an RSA cipher object and print the provider
final Cipher cipher =
Cipher.getInstance(ALGORITHM);
// encrypt the plain text using the public key
cipher.init(Cipher.ENCRYPT_MODE, key);
cipherText = cipher.doFinal(text.getBytes());
} catch (Exception e) {
e.printStackTrace();
}
return cipherText;
}

/**
* Decrypt text using private key.
*
* @param text
* :encrypted text
* @param key
* :The private key
* @return plain text
* @throws java.lang.Exception
*/
public static String decrypt(byte[] text, PrivateKey
key) {
byte[] dectyptedText = null;
try {
// get an RSA cipher object and print the provider
final Cipher cipher =
Cipher.getInstance(ALGORITHM);

// decrypt the text using the private key
cipher.init(Cipher.DECRYPT_MODE, key);
dectyptedText = cipher.doFinal(text);

} catch (Exception ex) {
ex.printStackTrace();
}
```

```
return new String(dectyptedText);
}
}

//Client

import java.io.*;
import java.net.*;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.NoSuchAlgorithmException;
import java.security.PrivateKey;
import java.security.PublicKey;

import javax.crypto.Cipher;

public class MyClientRsa
{public static final String
DES_ENCRYPTION_KEY = “testString”;
public static void main(String[] args)
{
try
{
Socket s=new Socket(“localhost”,6666);

DataOutputStream dout=new
DataOutputStream(s.getOutputStream());
BufferedReader br1 = new BufferedReader(new
InputStreamReader(System.in));
String input = br1.readLine();
System.out.println(“Enter message to encrypt”);
System.out.println(“Client Side”);
System.out.println(“Message from Client “+input);

// Check if the pair of keys are present else
generate those.
if (!Cryptography.areKeysPresent()) {
```

```
// Method generates a pair of keys using the RSA
algorithm and stores it
// in their respective files
Cryptography.generateKey();
}

// final String originalText = “Text to be encrypted
“;
ObjectInputStream inputStream = null;

// Encrypt the string using the public key
inputStream = new ObjectInputStream(new
FileInputStream(Cryptography.PUBLIC_KEY_FIL
E));
final PublicKey publicKey = (PublicKey)
inputStream.readObject();
final byte[] cipherText =
Cryptography.encrypt(input, publicKey);

// String encrypted = Cryptography.encrypt(input,
DES_ENCRYPTION_KEY);
System.out.println(“Encrypted
Message”+cipherText);

String encrypted =new String(cipherText);
dout.writeUTF(encrypted);
dout.flush();

dout.close();
s.close();

}
catch(Exception e)
{
System.out.println(e);
}

}
}
```

admin
on **July 17, 2013 at 8:28 pm** said:

> I feel you have missed to post the MyServerRsa class

Rahul
on **July 15, 2013 at 7:04 pm** said:

Superb tuto. thank you

I want to know that is there way to store these public and private key in software as String static final variable.

I dont have such secure environment to store private key. so i want to store key directly in code once key generated.

> **admin**
> on **July 15, 2013 at 7:25 pm** said:
>
> Yes you can create a static final variable for PublicKey and PrivateKey (instead of String) and use it.

Ghislain
on **August 18, 2013 at 6:11 pm** said:

Hello.
First of all, thanks for this tutorial and your code. But I've got an issue:
Your code works very well but I noticed that when I use the same private/public key, the encrypted text is not always the same. Of course, the decrypted text is always good, but I thought that when we use the RSA encryption, the ecrypted text is always the same, isn't it? Because with same keys and same text, you must have the same encrypted text. Can you explain

me why is it different in you code please?

Bye

> **admin**
> on **August 18, 2013 at 10:40 pm** said:
>
> Hi Ghislain,
>
> Thanks for visiting my blog.
> Yes, you are correct, with RSA same key and same text, the encrypted text should be the same.
> The reason why you see different encrypted text here is because of padding being used with RSA encryption.
> Some random text is added to the data before encryption and removed while decrypting.
> Padding is used to increase the security.
>
> Though we have not specified any padding in code, Cipher class will use some default padding.
>
> > Ghislain
> > on **August 19, 2013 at 3:40 am** said:
> >
> > OK thanks.
> > In fact I want to create an open source application for mobiles that aims to teach how RSA works. In the long term the application would create crypted text that you can send by SMS, mail … or whatever. I've learnt how RSA works and I think that in order to explain how it works the application has to encrypt an input text.
> > Do you think that I should add padding in the code, and if not, is it possible to have no padding added?
> > And again, thanks a lot for responding so quickly.
>
> > **admin**
> > on **August 20, 2013 at 12:24 am** said:
> >
> > Its always good to use padding as padding makes

encryption stronger and provides better security.
If you are wanting to create an open source application then you be using padding.

if you don't want to use padding you initialize the Cipher class using string "RSA/ECB/NoPadding".

zitkom
on **September 16, 2013 at 5:52 pm** said:

great tutorial. I have a question before, there is code:
public static final String PRIVATE_KEY_FILE =
"C:/keys/private.key";
"C:/keys/private.key" it's mean file location right? how if I use ubuntu?
sory for bad question, this is first time I learn encryption

admin
on **September 17, 2013 at 3:47 pm** said:

Thanks zitkom!
For Ubuntu or any other Linux OS, you can use the path "/home/keys/private.key"

admin
on **October 30, 2013 at 11:45 pm** said:

Hi Sumit,

Yes, you can see those files in the C:\keys folder.
There is no password to these files

Thanks for visiting the blog!

Sumit
on **October 21, 2013 at 3:50 pm** said:

Hi Admin, This tutorial was very useful.
I have small doubt that is public static final String PRIVATE_KEY_FILE = “C:/keys/private.key”;
public static final String PUBLIC_KEY_FILE = “C:/keys/public.key”; can i see the files ? what is the procedure for that ? What will be password for that file?

Ankaa... (@ittadiankarao)
on **December 30, 2013 at 12:13 pm** said:

Hi… Thanxs for your Valuable Replie to Other Users. I have already keys, so i am not generating them in code. I am able to do encryption and decryption. Samething implementing in c , But those outputs are not Same. Please help me any wrong in c code or java code…

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileReader;
import java.io.IOException;
import java.security.*;
import java.security.cert.*;
import java.security.cert.Certificate;
import java.util.Arrays;
import javax.crypto.*;
import org.bouncycastle.jce.provider.BouncyCastleProvider;
import org.bouncycastle.openssl.PEMReader;
import org.bouncycastle.openssl.PasswordFinder;

public class enc_rs {
public static final String ALGORITHM =
“RSA/ECB/PKCS1Padding”;
public static void main(String[] args) {
```

```
Security.addProvider(new BouncyCastleProvider());

try{

String filepath="/home/crl/Desktop/Cert/NewCert/RecPub.pem";
FileInputStream fin = new FileInputStream(filepath);
CertificateFactory f = CertificateFactory.getInstance("X.509");
Certificate certificate = f.generateCertificate(fin);
PublicKey pubKeyReceiver = certificate.getPublicKey();
final String originalText = "Central Research Laboratory";
final byte[] cipherText = encrypt(originalText, pubKeyReceiver);
System.out.println("Original Text: " + originalText);
System.out.println("Encrypted Text: " +cipherText.toString());

File privateKey = new
File("/home/crl/Desktop/Cert/NewCert/RecPvt.pem");
KeyPair keyPair = readKeyPair(privateKey,
"password".toCharArray());
final String plainText = decrypt(cipherText, keyPair.getPrivate());
System.out.println("Decrypted Text: " + plainText);
}

catch(Exception exp)
{
System.out.println(" Exception caught " + exp);
exp.printStackTrace();
}

}
public static byte[] encrypt(String text, PublicKey key) {
byte[] cipherText = null;
try {
// get an RSA cipher object and print the provider
final Cipher cipher = Cipher.getInstance(ALGORITHM);
// encrypt the plain text using the public key
cipher.init(Cipher.ENCRYPT_MODE, key);
cipherText = cipher.doFinal(text.getBytes());
} catch (Exception e) {
e.printStackTrace();
}
return cipherText;
}

public static String decrypt(byte[] text, PrivateKey key) {
byte[] dectyptedText = null;
try {
```

```
// get an RSA cipher object and print the provider
final Cipher cipher = Cipher.getInstance(ALGORITHM);

// decrypt the text using the private key
cipher.init(Cipher.DECRYPT_MODE, key);
dectyptedText = cipher.doFinal(text);

} catch (Exception ex) {
ex.printStackTrace();
}

return new String(dectyptedText);
}

private static class DefaultPasswordFinder implements
PasswordFinder {

private final char [] password;

private DefaultPasswordFinder(char [] password) {
this.password = password;
}

@Override
public char[] getPassword() {
return Arrays.copyOf(password, password.length);
}
}

private static KeyPair readKeyPair(File privateKey, char []
keyPassword) throws IOException {
FileReader fileReader = new FileReader(privateKey);
PEMReader r = new PEMReader(fileReader, new
DefaultPasswordFinder(keyPassword));
try {
return (KeyPair) r.readObject();
} catch (IOException ex) {
throw new IOException("The private key could not be decrypted",
ex);
} finally {
r.close();
fileReader.close();
}
}

}
```

C Code Implementation:

```
#include
#include
#include
#include
#include
//#include
#define UIDAI_PUBLIC_CERTIFICATE
“/home/crl/Desktop/Cert/NewCert/RecPub.pem”
#define UIDAI_PRIVATE_CERTIFICATE
“/home/crl/Desktop/Cert/NewCert/RecPvt.pem”

int assign_key_rsa(RSA *rsa, unsigned char *key, int n_len,
unsigned char *e_key, int e_len)
{
rsa->n = BN_bin2bn(key, n_len, rsa->n);
rsa->e = BN_bin2bn(e_key, e_len, rsa->e);
return 0;
}
void print_hex(unsigned char *out,int len)
{
int i;
printf(“\n”);
for(i=0;ipkey.rsa->n
BIGNUM *exp = epkey->pkey.rsa->e
pubKey = (unsigned char *)malloc(sizeof(unsigned char) *
bitSize);
unsigned char *eKey = (unsigned char *)malloc(sizeof(unsigned
char)*100);

int n_len = BN_bn2bin(publickey,pubKey);
int e_len = BN_bn2bin(exp,eKey);

memcpy(key,pubKey,n_len);
int i=0;
FILE *ex;
ex=fopen(“Keys.txt”,”w”);
printf(“\n————RSA PUBLIC KEY————\n”);
for(i=0;ipkey.rsa,RSA_PKCS1_PADDING);
outlen = len;
printf(“\n The Encrypted output “);
print_hex(outbuf,outlen);
printf(“\n%s “,outbuf);

printf(“\n Size: %d”,outlen);
free(pubKey);
```

```
EVP_PKEY_free(epkey);
X509_free(x);
RSA *rsapriv=0;
FILE *fp1;
fp1 = fopen(UIDAI_PRIVATE_CERTIFICATE,"rb");
if(fp1 == NULL)
{
printf(" NO UIDAI private Certificate found\n\n");
return -1;
}
outlen=rsapriv=(RSA *)PEM_read_RSAPrivateKey(fp1,&rsapriv,
(pem_password_cb *)"password",NULL);

RSA_private_decrypt(len,outbuf,in,rsapriv,RSA_PKCS1_PADDING);
RSA_free(rsapriv);
fclose(fp1);
printf("\n Decrypted Output");
print_hex(in,inlen);
printf("%s\n",in);
printf("\n Size: %d",inlen);
return 0;
}
```

saurabh
on **January 6, 2014 at 12:12 am** said:

THANK YOU!!! "REALLY AMAZING"

Bin
on **January 7, 2014 at 9:15 am** said:

Hi admin. i'm very appreciated for your trick. but i have a small mess:
You store a pair of key (included private.key) in the same location. I think it's not safe because hacker can also take it (private key), and decrypt if he successfully catch the encrypted

message.

> Bin
> on **January 7, 2014 at 9:18 am** said:
>
> P/s: could you explain me.
>
> > **admin**
> > on **January 7, 2014 at 6:12 pm** said:
> >
> > Thats correct, you cant store the public key and the private key at the same place.
> > This is just an example, not a real life scenario.
> >
> > Private key needs to kept secured.

codingatuni
on **January 15, 2014 at 5:49 pm** said:

Hey, first of all, thanks a lot for this awesome text. I was just wondering, when I run the program as is, I get the following output:

Original Text: Text to be encrypt
Encrypted Text: [B@33c26386
Decrypted Text: Text to be encrypt

Now this looks alright, but correct me if I'm wrong. The second output [B@33c26386 is just a location in memory, not what is actually stored there. What should I do if I wanted to do some further editing on the encrypted text ie. send it to someone? Again, thank you very much for this

> **admin**
> on **January 16, 2014 at 11:56 pm** said:

> You should be able to use the byte[] to send it to someone or do any further editing.
> I feel because I am calling the toString() method its printing the memory location, try Arrays.toString(cipherText) instead.

Brian Anderson
on **March 20, 2014 at 4:12 am** said:

admin,
Thanks for this well written and useful code.

What type of key files are the private.key and public.key files this code is generating? Are they binary DER-encoded format? If so, why does this OpenSSL command not work trying to convert it to a .pem file?
I tried
openssl x509 -inform def -in public.key -out public.pem
unable to load certificate
10684:error:0D07207B:asn1 encoding routines:ASN1_get_object:header too long:asn1_lib.c:150:

as well as
openssl pkcs12 -in public.key -out public.pem
8080:error:0D07207B:asn1 encoding routines:ASN1_get_object:header too long:asn1_lib.c:150:

I am trying to get the exponent and modulus from the public key to use in a Javascript encryption library and I think I could get them from a pem version of the key using openssl.

Thanks,
Brian

> **admin**
> on **March 22, 2014 at 3:39 am** said:
>
> The code is just writing the PrivateKey and PublicKey object to the file, not the actual keys.

If its needed to store the encoded private and public key to the file;
the getEncoded() method can be used; the method returns a byte[] which can be written to the file

This should resolve the the issue of converting:

```
FileOutputStream privateKeyOS = new
FileOutputStream(privateKeyFile);
privateKeyOS.write(key.getPrivate().getEncoded());
System.out.println( key.getPrivate());
System.out.println("Private key format " +
key.getPrivate().getFormat());
privateKeyOS.close();
```

Additionally, getFormat() method can be use to get the format of the public/private keys.
This public key here is using the PKCS#8 format.

If only the exponent and modulus of the keys is required, use print the PrivateKey/PublicKey object to the console, Following will give all the details of the private/public key pair:

```
System.out.println( key.getPrivate());
```

Durgesh Kumar
on **March 22, 2014 at 10:03 am** said:

Hi Sir the tutorial is very good but I have a question. How can we store that encrypted text into file & pass it to someone else. Or do further calculation on that text. Kindly assit me on it as I am stuck here. I want to encrypt some strings using this algo & decrypt it when needed.

admin
on **March 23, 2014 at 12:13 am** said:

Yes, its is possible to that, once text is encrypted, it can

be saved to a file or sent across to someone.
Once the set of private and public keys are generated.
It can be used that any to encrypt and decrypt data anytime.

chris
on **April 11, 2014 at 8:07 pm** said:

what key should i provide to someone. is it the private or the public?

> **admin**
> on **April 11, 2014 at 9:28 pm** said:
>
> Public key should be given to someone, which will be used to encrypt the data.
> Private will be used to decrypt the data.

sandip
on **April 14, 2014 at 11:18 pm** said:

hi… first of all thank you for the example. I am trying to develop a code that will encrypt a message and send it. will i be able to send the encrypted message?
if so how? please help

> **admin**
> on **April 15, 2014 at 12:06 am** said:
>
> Hey Sandip,
>
> Thanks for visiting my blog.

> Yes, you should be able to send it, just the way you would sent the normal text message.
> You only need encrypt before you send and decrypt once you receive it.

sandip
on **April 15, 2014 at 4:02 pm** said:

thanks for replying….
the output displays the memory location. so if i need to send the encrypted text will be need to send “cipherText” or “byte() cipherText”

> **admin**
> on **April 15, 2014 at 10:58 pm** said:
>
> cipherText .. as that is byte[] of the encrypted message.

sandip
on **April 15, 2014 at 11:08 pm** said:

thanks for helping.. it helped to clear my doubt.

i want to create 2 programs. one will create the public and private key and encrypt a message using the public key and save the cipher to a file. the second program will use the private key generated by the first program and decrypt the cipher. the first program is running thanks to you. i am storing the cipher as a txt file.. is this correct or should i use another file type?? how can i access the cipher from the file?? pls help

> **admin**

on **April 17, 2014 at 12:37 am** said:

Keys can be saved in a text file and keep in the the secured location, but is not a good approach.
This is just an example code.

If this needs to be used for practical purposes,
First get the encoded format of the keys using the getEncoded() method
Then write it into a file according to its format.

Please refer to:
http://docs.oracle.com/javase/tutorial/security/apisign/vstep2.html

sandip
on **April 15, 2014 at 11:12 pm** said:

related to my previous post

i need to access the cipher from the file in the 2nd program

sandip
on **April 21, 2014 at 1:17 pm** said:

thank u for ur last help. it helped me get a better understanding of encryption. i want to send an aes key through rsa encryption. here is what i am doing..
first i am generating the aes key, converting it into bytes by getencoded() function. do i need to convert the byte array into string and then encrypt it or do i directly encrypt the byte array???? after sending the key, it will be decrypted. after decryption i will either get the string or the byte array. how do i retreive the key???

admin
on **April 22, 2014 at 12:55 am** said:

I am sorry, I couldnt understand your questions correctly, but is what you will need to do
1. Generate the key pair, encoded and save the keys onto a file.
2. Read the file and decode the key.
3. Use the decoded key to encrypt or decrypt data.
4. If its required to send the key across, you can directly send the byte[].

abhishek patti (@abhipatti)
on **April 22, 2014 at 9:59 am** said:

Hi , Thank you so much for your code. I am writing simple RSA enabled Chat application in Java. My application works in following way I have a ChatServer.java which starts the server then I have ChatClient.java which contains code to generate the UI and RSA (your code) . First I run server then I run the client twice to generate two users(Two UI's) – Currently i generate keys when client is executed then i do encryption and send the message to another User in encrypted form. How to generate separate keys for each user and how to broadcast each users public key ?

Here is my code

Chatserver.java

```
// Chat Server runs at port no. 9999
import java.io.*;
import java.util.*;
import java.net.*;
import static java.lang.System.out;

public class ChatServer {
Vector users = new Vector();
Vector clients = new Vector();

public void process() throws Exception {
```

```
ServerSocket server = new ServerSocket(9999,10);
out.println("Server Started…");
while( true) {
Socket client = server.accept();
HandleClient c = new HandleClient(client);
clients.add(c);
} // end of while
}
public static void main(String … args) throws Exception {
new ChatServer().process();
} // end of main

public void boradcast(String user, String message) {
// send message to all connected users
for ( HandleClient c : clients )
if ( ! c.getUserName().equals(user) )
c.sendMessage(user,message);
}

class HandleClient extends Thread {
String name = "";
BufferedReader input;
PrintWriter output;

public HandleClient(Socket client) throws Exception {
// get input and output streams
input = new BufferedReader( new InputStreamReader(
client.getInputStream())) ;
output = new PrintWriter ( client.getOutputStream(),true);
// read name
name = input.readLine();
users.add(name); // add to vector
start();
}

public void sendMessage(String uname,String msg) {
output.println( uname + ":" + msg);
}

public String getUserName() {
return name;
}
public void run() {
String line;
try {
while(true) {
```

```
line = input.readLine();
if ( line.equals(“end”) ) {
clients.remove(this);
users.remove(name);
break;
}
boradcast(name,line); // method of outer class – send messages
to all
} // end of while
} // try
catch(Exception ex) {
System.out.println(ex.getMessage());
}
} // end of run()
} // end of inner class

} // end of Server
```

CHatclient.java

```
import java.io.*;
import java.util.*;
import java.net.*;

import javax.swing.*;

import java.awt.*;
import java.awt.event.*;

import static java.lang.System.out;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.NoSuchAlgorithmException;
import java.security.PrivateKey;
import java.security.PublicKey;

import javax.crypto.Cipher;
```

```
public class ChatClient extends JFrame implements
ActionListener {
String uname;
PrintWriter pw;
BufferedReader br;
JTextArea taMessages;
JTextField tfInput;
JButton btnSend,btnExit;
Socket client;

/**
* String to hold name of the encryption algorithm.
*/
public static final String ALGORITHM = “RSA”;

/**
* String to hold the name of the private key file.
*/
public static final String PRIVATE_KEY_FILE =
“C:/keys/private.key”;

/**
* String to hold name of the public key file.
*/
public static final String PUBLIC_KEY_FILE =
“C:/keys/public.key”;

/**
* Generate key which contains a pair of private and public key
using 1024
* bytes. Store the set of keys in Prvate.key and Public.key files.
*
* @throws NoSuchAlgorithmException
* @throws IOException
* @throws FileNotFoundException
*/
public static void generateKey() {
try {
final KeyPairGenerator keyGen =
KeyPairGenerator.getInstance(ALGORITHM);
keyGen.initialize(1024);
final KeyPair key = keyGen.generateKeyPair();

File privateKeyFile = new File(PRIVATE_KEY_FILE);
File publicKeyFile = new File(PUBLIC_KEY_FILE);
```

```
// Create files to store public and private key
if (privateKeyFile.getParentFile() != null) {
privateKeyFile.getParentFile().mkdirs();
}
privateKeyFile.createNewFile();

if (publicKeyFile.getParentFile() != null) {
publicKeyFile.getParentFile().mkdirs();
}
publicKeyFile.createNewFile();

// Saving the Public key in a file
ObjectOutputStream publicKeyOS = new ObjectOutputStream(
new FileOutputStream(publicKeyFile));
publicKeyOS.writeObject(key.getPublic());
publicKeyOS.close();

// Saving the Private key in a file
ObjectOutputStream privateKeyOS = new ObjectOutputStream(
new FileOutputStream(privateKeyFile));
privateKeyOS.writeObject(key.getPrivate());
privateKeyOS.close();
} catch (Exception e) {
e.printStackTrace();
}

}

/**
* The method checks if the pair of public and private key has
been generated.
*
* @return flag indicating if the pair of keys were generated.
*/
public static boolean areKeysPresent() {

File privateKey = new File(PRIVATE_KEY_FILE);
File publicKey = new File(PUBLIC_KEY_FILE);

if (privateKey.exists() && publicKey.exists()) {
return true;
}
return false;
}

/**
```

```
* Encrypt the plain text using public key.
*
* @param text
* : original plain text
* @param key
* :The public key
* @return Encrypted text
* @throws java.lang.Exception
*/
public static byte[] encrypt(String text, PublicKey key) {
byte[] cipherText = null;
try {
// get an RSA cipher object and print the provider
final Cipher cipher = Cipher.getInstance(ALGORITHM);
// encrypt the plain text using the public key
cipher.init(Cipher.ENCRYPT_MODE, key);
cipherText = cipher.doFinal(text.getBytes());
} catch (Exception e) {
e.printStackTrace();
}
return cipherText;
}

/**
* Decrypt text using private key.
*
* @param text
* :encrypted text
* @param key
* :The private key
* @return plain text
* @throws java.lang.Exception
*/
public static String decrypt(byte[] text, PrivateKey key) {
byte[] dectyptedText = null;
try {
// get an RSA cipher object and print the provider
final Cipher cipher = Cipher.getInstance(ALGORITHM);

// decrypt the text using the private key
cipher.init(Cipher.DECRYPT_MODE, key);
dectyptedText = cipher.doFinal(text);

} catch (Exception ex) {
ex.printStackTrace();
}
```

```
return new String(dectyptedText);
}

public ChatClient(String uname,String servername) throws
Exception {
super(uname); // set title for frame
this.uname = uname;
client = new Socket(servername,9999);
br = new BufferedReader( new InputStreamReader(
client.getInputStream()) ) ;
pw = new PrintWriter(client.getOutputStream(),true);
pw.println(uname); // send name to server
buildInterface();
new MessagesThread().start(); // create thread for listening for
messages
}

public void buildInterface() {
btnSend = new JButton("Send");
btnExit = new JButton("Exit");
taMessages = new JTextArea();
taMessages.setRows(10);
taMessages.setColumns(50);
taMessages.setEditable(false);
tfInput = new JTextField(50);
JScrollPane sp = new JScrollPane(taMessages,
JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED,
JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
add(sp,"Center");
JPanel bp = new JPanel( new FlowLayout());
bp.add(tfInput);
bp.add(btnSend);
bp.add(btnExit);
add(bp,"South");
btnSend.addActionListener(this);
btnExit.addActionListener(this);
setSize(500,300);
setVisible(true);
pack();
}

@SuppressWarnings("resource")
public void actionPerformed(ActionEvent evt) {
String originalText = tfInput.getText();
ObjectInputStream inputStream = null;
try {
```

```
inputStream = new ObjectInputStream(new
FileInputStream(PUBLIC_KEY_FILE));
} catch (FileNotFoundException e) {
// TODO Auto-generated catch block
e.printStackTrace();
} catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
PublicKey publicKey = null;
try {
publicKey = (PublicKey) inputStream.readObject();
} catch (ClassNotFoundException e) {
// TODO Auto-generated catch block
e.printStackTrace();
} catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
final byte[] cipherText = encrypt(originalText, publicKey);
if ( evt.getSource() == btnExit ) {
pw.println("end"); // send end to server so that server know about
the termination
System.exit(0);
} else {
// send message to server
pw.println(cipherText.toString());
}
}

public static void main(String … args) {

// Check if the pair of keys are present else generate those.
if (!areKeysPresent()) {
// Method generates a pair of keys using the RSA algorithm and
stores it
// in their respective files
generateKey();
}
// take username from user
String name = JOptionPane.showInputDialog(null,"Enter your
name :", "Username",
JOptionPane.PLAIN_MESSAGE);
String servername = "localhost";
try {
new ChatClient( name ,servername);
```

```
} catch(Exception ex) {
out.println( “Error –> ” + ex.getMessage());
}

} // end of main

// inner class for Messages Thread
class MessagesThread extends Thread {
public void run() {
String line;
try {
while(true) {
line = br.readLine();
taMessages.append(line + “\n”);
} // end of while
} catch(Exception ex) {}
}
}
} // end of client
```

Priyanka
on **April 27, 2014 at 1:22 am** said:

Can i get the GUI as well, along with the initial encryption decryption code?? Like how to select the files to encrypt, where to save it. Using swing awt. I am not getting. I’d be glad if anyone could help, as i am doing my minor proj on rsa application.

Ric
on **July 20, 2014 at 4:17 pm** said:

Hi, i attempted to save the 2 keys on a database and retrieved them out. And the encryption seems to work but i cant decrypt the cipher text back to plain text.

i passed the value from the database to the eclipse [the values are still the same and correct]
reconverted the value back to public and private keys.
i tried encrypting the input text and cipherText.toString() returns [B@528b73d0
but decrypting it returned ⍰2⍰,n⍰⍰T⍰⍰⍰qW!}C⍰⍰⍰ s⍰⍰⍰_R⍰52ٟ⍰n^NT⍰%F⍰(+7ش><⍰O⍰⍰H:><⍰⍰T⍰ r⍰ +⍰,c

did i do something wrong? or the conversion broke the codes somehow?

```
JSONObject json = userFunction.retrieveKey(recipients);
JSONObject json_user = json.getJSONObject("user");
String pubkeyStr = json_user.getString(KEY_PUBLIC);
final byte[] finalKeyByte = stringToByte(pubkeyStr);

final PublicKey keyFinales = backToKey(finalKeyByte);
final byte[] cipherText = encrypt(bodys, keyFinales);
final String bodyText = byteToString(cipherText);

final byte[] recievedText = stringToByte(bodyText);
String privkeyStr = json_user.getString(KEY_PRIVATE);
final byte[] finalPKeyByte = stringToByte(privkeyStr);
final PrivateKey keyFinalese = backToPKey(finalPKeyByte);
final String plainText = decrypt(recievedText, keyFinalese);
```

> Ric
> on **July 20, 2014 at 4:49 pm** said:
>
> nvm.. i figured the error

thandar
on **August 30, 2014 at 10:24 am** said:

Hi, I want to encrypt with certificate private key and decrypt with public key.
Is there any sample? Can give advice? Please!
Thank u

AceG
on **November 2, 2014 at 10:38 pm** said:

What would the original code resemble if I wanted to fed the algorithm with two prime numbers p and q along with e = “65537”, where e is an exponential part of the public key.

Mohammed
on **February 9, 2015 at 9:24 pm** said:

THANK YOU SO MUCH FOR THIS WONDERFUL PIECE.

Ayan Chandra
on **May 29, 2015 at 7:02 pm** said:

Hello Guys,
Just need your extended help. I am following the whole code and trying to implement same in unix. Only change I have to make is store the Encrypted string in a file in the form of Key value pair. Now when I am decrypting it doFinal is failing stating following error.

javax.crypto.BadPaddingException: Decryption error
at com.ibm.crypto.provider.RSAPadding.c(Unknown Source)
at com.ibm.crypto.provider.RSAPadding.unpad(Unknown Source)
at com.ibm.crypto.provider.RSACipher.a(Unknown Source)
at com.ibm.crypto.provider.RSACipher.engineDoFinal(Unknown Source)
at javax.crypto.Cipher.doFinal(Unknown Source)
at DecryptionUtil.decrypt(DecryptionUtil.java:103)

at DecryptionUtil.main(DecryptionUtil.java:151)
java.lang.NullPointerException
at java.lang.String.(String.java:123)
at DecryptionUtil.decrypt(DecryptionUtil.java:108)
at DecryptionUtil.main(DecryptionUtil.java:151)

> **admin**
> on **May 31, 2015 at 4:43 pm** said:
>
> Try these:
>
> A. public static final String ALGORITHM = “RSA/ECB/PKCS1Padding”
> B. while encrypting: cipherText = cipher.doFinal(text.getBytes(StandardCharsets.UTF_8));
> C. while decrypting: return new String(dectyptedText, StandardCharsets.UTF_8);