The Bouncy castle

What is it?

The Bouncy Castle Crypto package is a Java implementation of cryptographic algorithms and it was developed by the Legion of the Bouncy Castle. The package is organised so that it contains a light-weight API suitable for use in any environment (including J2ME) with the additional infrastructure to conform the algorithms to the JCE framework.

Address, version, license

Packages (both binary and source code) are avaiable on http://www.bouncycastle.org/.

Licence of The Bouncy castle is quite simillar to licence of OpenSSL so you're allowed to use it even in commercial products. Details about licence can be found on URL: http://www.bouncycastle.org/licence.html

Some of the algorithms in the Bouncy Castle APIs are patented in some places. It is upon the user of the library to be aware of what the legal situation is in their own situation. Examples of such algorithm is IDEA which is patented in the USA, Japan, and Europe including at least Austria, France, Germany, Italy, Netherlands, Spain, Sweden, Switzerland and the United Kingdom. Non-commercial use is free, however any commercial products that make use of IDEA are liable for royalties.

Current stable version of Java library 1.41.

Documentation

Documentation for Java version of the library can be found in standard Javadoc format. Source packages contains many good examples how to work with this library, so if you're intrested, you can find examples how to use the BC by JCE or how to use light weight API (which seems to be sometimes more usefull).

Advantages/Disadvantages

On of the advantages of this library is provided light-weight API which allows programmer use cryptographic functions without using JCE framework.

This library also supports reading and writing encoded ASN.1 objects, provides a light weight client-side TLS API, support for generating X.509 certificates (version 1,2,3), working with S/MIME and many more.

Disadvantage is if you're trying to do something unusual it's really hard to do it using The Bouncy castle.

Avaiable cryptographic algorithms

Library itself implements all cryptographic algorithms as SunJCE (default JCE provider for J2SDK5), but also implements some other cryptographic algorithms (AESWrap, CAST6, ECIES, ...). The BC even supports elliptic curves (even there are currently no classes to support EC in JCE or JCA).

List of all provided cryptographic algorithms can be found on URL:
http://www.bouncycastle.org/specifications.html

Using the library

If you want to use The BC as JCE provider, you need to register it as one of the JCE providers (default JCE provider in Java is SunJCE). Short tutorial how to setup JVM to use The Bouncy castle as JCE provider can be find on URL:
http://www.bouncycastle.org/wiki/display/JA1/Provider+Installation.

If you want to register provider dynamically you need to add those lines to your program:

```
import org.bouncycastle.jce.provider.BouncyCastleProvider;
...
Security.addProvider(new BouncyCastleProvider());
```

and to install in statically you need to modify file *java.security* and add following line:

```
security.provider.N=org.bouncycastle.jce.provider.BouncyCastleProvider
```

where N should be as low as it's possible (but setting it to 1 isn't also good idea).

Source code - example

Following source code is part of encrypting rutine which is using DES algorithm to encrypt String *inputString* using key passed in String *keyString*.

```
/*
 * This will use a supplied key, and encrypt the data
 * This is the equivalent of DES/CBC/PKCS5Padding
*/
BlockCipher engine = new DESEngine();
BufferedBlockCipher cipher = new PaddedBlockCipher(new CBCCipher(engine));

byte[] key = keyString.getBytes();
byte[] input = inputString.getBytes();

cipher.init(true, new KeyParameter(key));
```

```
byte[] cipherText = new byte[cipher.getOutputSize(input.length)];

int outputLen = cipher.processBytes(input, 0, input.length, cipherText, 0);
try
{
 cipher.doFinal(cipherText, outputLen);
 }
 catch (CryptoException ce)
 {
     System.err.println(ce);
     System.exit(1);
 }
```