Introduction to Bouncy Castle APIs

The Bouncy Castle Crypto APIs is a set of implementations of cryptographic concepts and algorithms. Bouncy Castle APIs are developed in both Java and C#. These APIs can be freely downloaded from the Bouncy Castle home page.

Let us have a glance at some of the important type of implementations and algorithms provided by BC APIs -

Digests

Digests are the hash values which are generated after applying a hash algorithm on message. Digests are also known as Hash Value, fingerprints, checksums, etc.

Bouncy castle APIs provide us, implementation of following hash algorithms –

- GOST3411
- MD2, MD4, MD5
- RIPEMD128, RIPEMD160, RIPEMD256, RIPEMD320
- SHA-1, SHA-224, SHA-256, SHA-384, SHA-512
- Tiger
- Whirlpool.

X.509

X509 is a standard, which specifies the standard format for public key certificates, Certificate revocation list and attribute certificates. Bouncy Castle APIs include the implementation for generators, parsers for Version1 and Version3 certificates, Version 2 CRLs and attribute certificates.

It provides the classes for X509 formats, also, e.g. PEM, p12, pfx etc.

Following are some of the important X509 related classes, provided by Bouncy Castle –

- PEMParser
- X509 certficate
- X509 certficate parser
- X509 crl
- X509 signature util
- X509 v1 generator
- X509 v2 generator
- X509 v3 generator

Password Based Encryption

There are many instances when data should be encrypted by the passphrase or password provided by the user. This secret password selected by the user acts like the encryption key. This is unlike the situation in which the keys are generated by the system itself. However, there are steps to be followed for generating or deriving the key from the password selected by the user.

- A hash value of the password text is computed. This is done by using the hash algorithms like SHA1. To make the derived key strong, salt is appended in the password. (Salts are the extra random bytes added in the message during encryption. Salt is different and random for every input, thus the output produced by the hash algorithms also differ every time and hence dictionary attacks are prevented.
- A DES or AES key is generated from the password and salt.

Bouncy castle provides the following PBE algorithms –

- SHA1 and DES-CBC,
- SHA1 and RC2-CBC,
- SHA-1 and 128bit RC4,
- SHA-1 and 40bit RC4,
- SHA-1 and 3-key DESEDE-CBC,
- SHA-1 and 2-key DESEDE-CBC,
- SHA-1 and 128bitRC2-CBC,
- SHA-1 and 40bit RC2-CBC,
- HmacSHA-1,
- Hmac SHA-224,
- HmacSHA-256,
- HmacRIPEMD128,
- HmacRIPEMD160,
- HmacRIPEMD256.

Signature algorithms

Signature are applied to digital data to verify that the data is not tampered during transmission and also, that the data has been sent by the correct authority or person.

In signature mechanism, the hash value of the message is computed, using the Hash algorithms like MD2, MD5 etc. This hashed value is used by the signature algorithm to generate the digital signature. Note: - for signing, the private key is needed by the algorithm. The digital signature is sent along with the message.

Following implementation are provided by the Bouncy castle apis-

- MD2withRSA,
- MD4withRSA,
- MD5withRSA,
- RIPEMD128withRSA,
- RIPEMD160withRSA,
- RIPEMD256withRSA,
- SHA-1withRSA,
- SHA-224withRSA,
- SHA-256 with RSA and MGF1,
- SHA-384 with RSA and MGF1,
- SHA-512 with RSA and MGF1,
- SHA-1 with DSA,
- SHA-1 with ECDSA.

Symmetric key algorithms:

Following secret key algorithm are implemented –

- AES
- Blowfish
- Camellia
- CAST5, CAST6
- DESede, DES
- GOST28147
- HC-128, HC-256

- IDEA
- NaccacheStern
- RC2, RC4, RC5-32, RC5-64, RC6,
- Rijndael, Serpent, Skipjack, TEA/XTEA, Twofish, and VMPC.

Asymmetric key algorithms:

Following asymmetric key algorithm are implemented -

- RSA (with blinding)
- ElGamal
- DSA
- ECDSA.

Apart from the above listed algorithms, Bouncy Castle provides implementation for PGP (Pretty good Privacy, Online Certificate Status Protocol. Time Stamp Protocol etc.

Source: http://www.go4expert.com/articles/introduction-bouncy-castle-apis-t24710/