Java Cryptography API

The Java Cryptography API enables you to encrypt and decrypt data in Java, as well as manage keys, sign and authenticate messages, calculate cryptographic hashes and much more.

Java Cryptography Architecture

The Java Cryptography Architecture (JCA) is the name for the internal design of the Java cryptography API. JCA is structured around some central general purpose classes and interfaces.
The Java cryptography API is divided between the following Java packages:

- java.security
- java.security.cert
- java.security.spec
- java.security.interfaces
- javax.crypto
- javax.crypto.spec
- javax.crypto.interfaces

Some of the core classes and interfaces of these packages are:

- Provider
- Cipher
- KeyFactory
- SecretKeyFactory
- KeyPairGenerator
- KeyGenerator
- CertificateFactory
- CertStore

Provider

The Provider (java.security.Provider) class is a central class in the Java cryptography API. In order to use the Java crypto API you need a Provider set. One of the most popular cryptography providers for the Java cryptography API is called **Bouncy Castle.**

Cipher

The Cipher (javax.crypto.Cipher) class represents a cryptographic algorithm. A cipher can be used to both encrypt and decrypt data.

```
Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
```

where,
AES is the encryption algorithm
CBC part is a mode the AES algorithm can work in.
PKCS5Padding part is how the AES algorithm should handle the last bytes of the data.

Keys

To encrypt or decrypt data you need a key. There are two types of keys - depending on the encryption algorithm:

- Symmetric keys
- Asymmetric keys

Symmetric keys are used for symmetric encryption algorithms. Symmetric encryption algorithms use the same key for encryption and decryption.

Asymmetric keys are used for asymmetric encryption algorithms. Asymmetric encryption algorithms use one key for encryption, and another for decryption. The public key - private key encryption algorithms are examples of asymmetric encryption algorithms.