Java Cryptography Architecture

Design principles

✓ Implementation independence
  ✓ Applications do not need to implement security algorithms
  ✓ Services are implemented in providers pluggable into the Java platform

✓ Implementation interoperability - providers are interoperable

✓ Algorithm extensibility
Architecture
Providers

✓ `java.security.Provider` - base class of all providers
  ✓ advertises algorithms
  ✓ Supply concrete implementations

```java
md = MessageDigest.getInstance("MD5")
md = MessageDigest.getInstance("MD5", "ProviderC")
```
Engine Classes

✓ Provide the interface to a specific type of cryptographic service
  ✓ cryptographic operations
  ✓ generators or converters of cryptographic material
  ✓ objects (keystores or certificates)
Example Engine Classes

- SecureRandom
- MessageDigest
- Cipher
- Signature
- MAC
- KeyStore
import javax.crypto.*;
Cipher c = Cipher.getInstance("AES");
c.init(ENCRYPT_MODE, key);
SecureRandom

✓ Seeding
synchronized public void setSeed(byte[] seed)
public void setSeed(long seed)

✓ Using the object
synchronized public void nextBytes(byte[] bytes)

✓ Generate seed bytes
byte[] generateSeed(int numBytes)
MessageDigest

- Updating the object
  
  `void update(byte input)`
  `void update(byte[] input)`

- Computing the digest
  
  `byte[] digest()`
  `byte[] digest(byte[] input)`
Your Best Friend

✓ Look up API docs for the relevant packages
  ✓ `java.security`
  ✓ `javax.crypto`

✓ JCA reference guide