Java Cryptography Architecture
Java Cryptography Extension
JCA/JCE

J. Bacelar Almeida
jba@di.uminho.pt

Departamento de Informática
Universidade do Minho

2008/2009

Princípios

- *Framework* do Java que fornece uma API para *serviços criptográficos*.
- Adopta uma arquitectura que lhe permite ser:
  - **Extensivel:** novas funcionalidades podem ser incorporadas;
  - **Independente das Implementações:** suporta diferentes implementações (*Providers*) para os serviços disponíveis (e essas diferentes implementações podem coabitar num dado programa).
  - **Independente das Técnicas:** organiza diferentes técnicas em função da funcionalidade (orientada ao *serviço*).
- Distinção entre JCA e JCE é meramente histórica (questões legais relativas à exportação de tecnologia criptográfica dos EUA).

Serviços (*Engine Classes*)

- **Cipher** – cifra.
- **CipherInputStream/CipherOutputStream/SealedObject** – abstracções de alto nível sobre objectos de cifra.
- **SecureRandom** – gerador de números aleatórios seguro.
- **KeyGenerator/KeyPairGenerator** – geradores de chaves.
- **KeyFactory** – conversão de formatos para chaves.
- **KeyAgreement** – acordo de chaves.
- **AlgorithmParameterGenerator** – gerador de parâmetros (a serem passados a outros objectos, como cifras...).
- **MessageDigest/MAC** — funções de hash criptográficas, MACs.
- **Signature** — assinaturas digitais.
- **X509Certificate** — certificados X509.
- **...**

Utilização

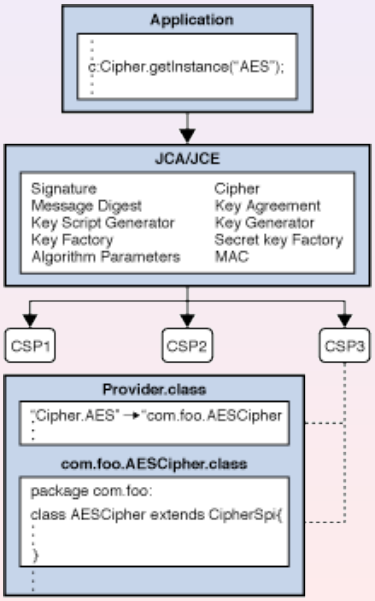
- A utilização dos serviços criptográficos segue um padrão bem definido:
  - **Crição de instância** – *requisição do serviço* – através do método estático `getInstance` (*factory method*). O utilizador especifica qual o algoritmo pretendido e o *framework* selecciona a implementação apropriada (*provider*).
  - **Inicialização** — tipicamente através do método `init`.
  - **Utilização** — através de métodos específicos de cada *engine class*: `update`, `doFinal`, `sign`, etc.

Exemplo

```
// Cria instância da cifra
Cipher e = Cipher.getInstance("RC4");

// Cria instância do gerador de chaves
KeyGenerator kg = KeyGenerator.getInstance("RC4");

// Inicializa gerador de chaves (128bit) e gera chave
kg.init(128);
SecretKey key = kg.generateKey();

//Alternativa...
//byte[] key_bytes = { 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x0
//SecretKey key = new SecretKeySpec(key_bytes,"RC4");

// Inicializa cifra
e.init(Cipher.ENCRYPT_MODE,key);

// Utiliza a cifra (in e out são arrays de bytes
//   ou streams)
e.doFinal(in, out);
```

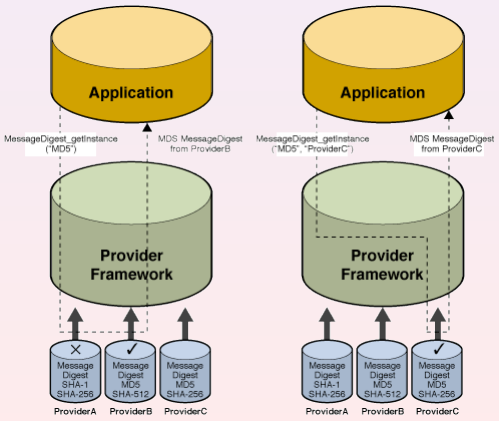

Engine Classes vs Implementações

- A independência das implementações é garantida impondo que o acesso a “implementações concretas” seja sempre realizado pela *engine class* respectiva.



Selecção de Implementações

- Quando o utilizador solicita um algoritmo de uma *engine class* específica, o *framework* irá seleccionar uma implementação concreta de um dos *providers* disponíveis (segundo uma ordem pré-seleccionada);
- Opcionalmente, pode ser pedida uma implementação de um *provider* específico.



Extensibilidade

- Novos *providers* podem ser adicionados ao sistema;
- Esses *providers* podem fornecer implementações para novos algoritmos ou implementações alternativas dos algoritmos existentes.