# Segurança da Informação

**Competências:**

- Implementar práticas e condutas de segurança da informação no ambiente de TI.

**Bases Tecnológicas, científicas e instrumentais (conteúdos):**

- Arquitetura de Criptografia Java;

**Situação de Aprendizagem:**

- Arquitetura de Criptografia Java.

# Criptografia com Java


- JCA – Java Cryptography Architecture
  - framework de segurança integrado com o core do Java. ;
  - java.security;

- JCE – Java Cryptography Extension
  - extensões para criptografia, essencialmente; e
  - javax.crypto.

# JCA - Engines

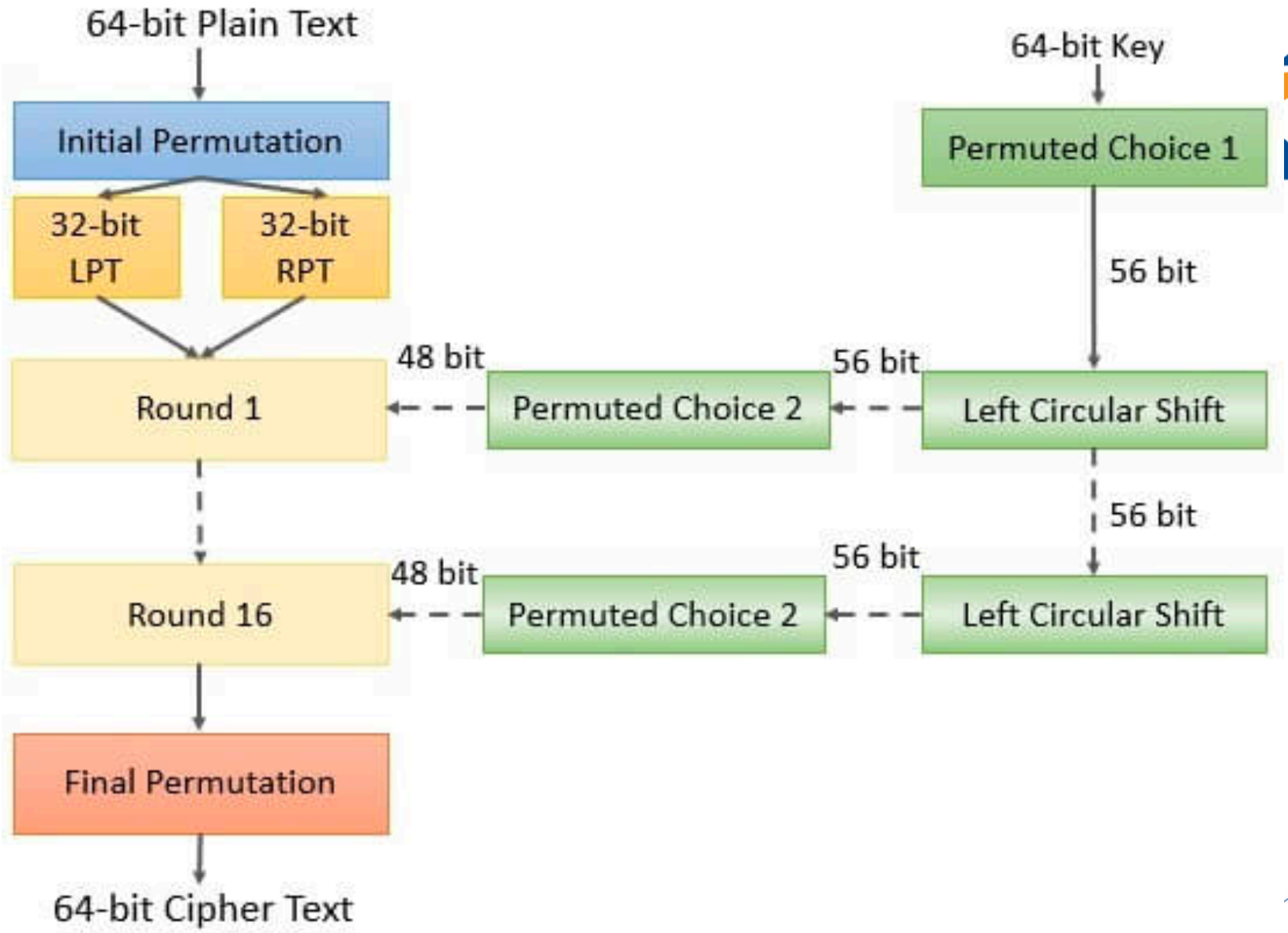
- Provê uma interface para um tipo específico de serviço

- Serviços fornecidos:
    a) MessageDigest
    b) Signature
    c) KeyFactory
    d) KeyStore
    e) SecureRandom
    f) CertificateFactory
    g) CertPathBuilder
    h) CertStore

# JCE - Engines

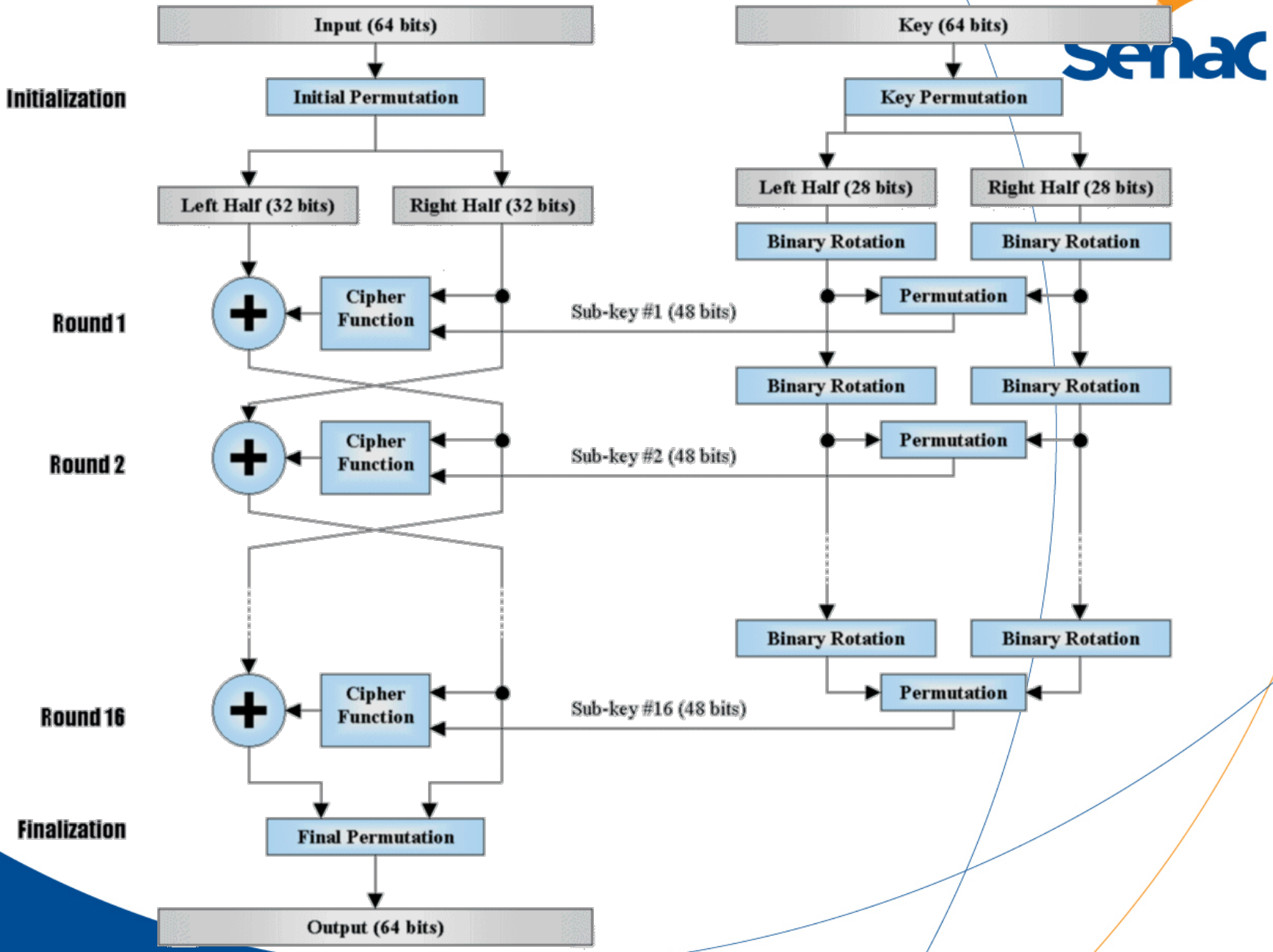
- Provê uma interface para um tipo específico de serviço

- Serviços fornecidos:
    a) Criptografia
    b) keyGenerator
    c) SecretKeyFactor
    d) KeyAgreement
    e) Mac




Data Encryption Standard (DES)

# Exemplo


Input (64 bits)
Key (64 bits)
Initialization
Initial Permutation
Key Permutation
Left Half (32 bits)
Right Half (32 bits)
Left Half (28 bits)
Right Half (28 bits)
Binary Rotation
Binary Rotation
Round 1
Cipher Function
Sub-key #1 (48 bits)
Permutation
Binary Rotation
Binary Rotation
Round 2
Cipher Function
Sub-key #2 (48 bits)
Permutation
Binary Rotation
Binary Rotation
Round 16
Cipher Function
Sub-key #16 (48 bits)
Permutation
Finalization
Final Permutation
Output (64 bits)

# Exemplo

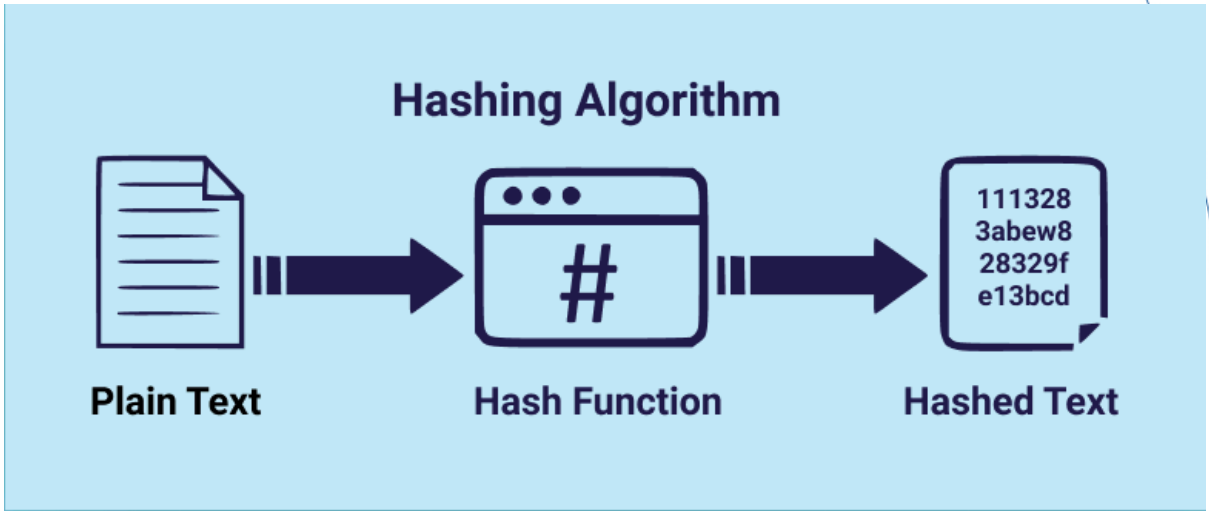
## criptografia

```
byte[] textoEncriptado(SecretKey desKey) throws NoSuchPaddingException,
NoSuchAlgorithmException, IllegalBlockSizeException, BadPaddingException,
InvalidKeyException {
    byte[] message = "The quick brown fox jumps over the lazy dog".getBytes();
    Cipher desCipher = Cipher.getInstance("DES/ECB/PKCS5Padding");
    desCipher.init(Cipher.ENCRYPT_MODE, desKey);
    byte[] encryptedMessage = desCipher.doFinal(message);
    return encryptedMessage;
}
```

```
String textoClaro(byte[] cipher, SecretKey desKey) throws NoSuchPaddingException,
NoSuchAlgorithmException, IllegalBlockSizeException, BadPaddingException,
InvalidKeyException {
    Cipher desCipher = Cipher.getInstance("DES/ECB/PKCS5Padding");
    desCipher.init(Cipher.DECRYPT_MODE, desKey);
    String plainMessage = new String(desCipher.doFinal(cipher), StandardCharsets.UTF_8);
    return plainMessage;
}
```

# Exemplo

## hash




```
SecureRandom secRandom = SecureRandom.getInstance("SHA1PRNG");
secRandom.setSeed(711);
byte[] bytes = new byte[20];
secRandom.nextBytes(bytes);
```

# Exercício

- Implemente uma aplicação que faça a criptografia do texto:

É madeira de vento, tombo da ribanceira
É o mistério profundo, é o queira ou não queira
É o vento vetando, é o fim da ladeira
É a viga, é o vão, festa da cumeeira
É a chuva chovendo, é conversa ribeira
Das águas de março, é o fim da canseira
É o pé, é o chão, é a marcha estradeira
Passarinho na mão, pedra de atiradeira

- Atendendo os seguintes requisitos:
  - Para encriptar:
    - O texto em claro deverá estar em um arquivo chamado **aguasdemarco.txt**;
    - O nome do arquivo é fornecido através da linha de comando (argumento String Args[] do método main);
    - O algoritmo a ser utilizado é o AES;
    - O texto criptografado deverá ser armazenado em um arquivo chamado **aguasdemarco.cripto**;
  - Para decriptar:
    - O texto criptografado deverá estar em um arquivo chamado **aguasdemarco.cripto**;
    - O nome do arquivo é fornecido através da linha de comando (argumento String Args[] do método main);
    - O algoritmo a ser utilizado é o AES;
    - O texto criptografado deverá ser armazenado em um arquivo chamado **aguasdemarcoemclaro.txt**;

# Semana que vem Avaliação do Ciclo 2