Mã hóa với JAVA

Phạm Nguyên Khang

Java Cryptography Architecture - JCA

- Xuất hiện từ JDK1.1, được chuẩn hóa từ JDK1.4 gồm các gói:
  - java.security
  - javax.crypto
  - javax.security
- Cung cấp khoảng 11 lớp cần thiết đảm bảo an toàn cho các ứng dụng Java
- Đề nghị các cơ chế bảo mật từ thấp đến cao
  - Mã hóa, hàm băm
  - Chữ ký điện tử, chứng chỉ
  - SSL, HTTPS
  - Chứng thực và điều khiển truy cập

Nguyên lý cơ bản

- Độc lập với cài đặt
  - Alice có thể sử dụng bản cài đặt DES của mình để giải mã thông điệp được mã hóa bằng bản cài đặt của Peter
- Độc lập với giải thuật và khả năng mở rộng
  - Nếu Alice và Peter quyết định sử dụng giải thuật TripleDES thay vì DES thì họ chỉ cần đổi **tên của giải thuật** trong chương trình của họ. Các phần khác vẫn không thay đổi.

Độc lập với cài đặt

- Nhà cung cấp dịch vụ mã hóa (cryptographic service provider) cung cấp các gói dịch vụ như: mã hóa, các giải thuật chữ ký điện tử, ...
  - Alice sử dụng giải thuật "TripleDES" của nhà cung cấp dịch vụ "AliceCrypto" để mã hóa các thông điệp của cô.

- Nhà cung cấp mặc định: SUN

Một số lớp quan trọng

- Các dịch vụ mã hóa được định nghĩa trừu tượng bằng các "lớp động cơ" (engine classes)
  - Cipher: mã hóa
  - MessageDigest: băm
  - Signature: chữ ký điện tử
  - KeyFactory: sinh khóa
  - SecureRandom: sinh số ngẫu nhiên an toàn
  - …

Tạo đối tượng (Instanciation)

- Sử dụng các "hàm sản suất" (factory methods) getInstance(…) của các "lớp động cơ"
- Có thể tạo các đối tượng sử dụng một giải thuật cụ thể của một nhà cung cấp dịch vụ mã hóa nào đó.
  - Để tạo ra đối tượng có khả năng mã hóa với giải thuật DES của nhà cung cấp dịch vụ mặc định (SUN), ta sử dụng hàm sản xuất của lớp Cipher:
    - Cipher.getInstance("DES");
  - Để sử dụng giải thuật DES của nhà cung cấp AliceCrypto:
    - Cipher.getInstance("DES", "AliceCrypto");

Mã hóa

- Lớp động cơ: Cipher
  - Tạo ra các bộ mã hóa/ giải mã cho một giải thuật và một khóa cho trước
- Các lớp sinh khóa (KeyGenerator) và xây dựng khóa (KeyFactory) chịu trách nhiệm tạo ra các khóa cần thiết cho việc mã hóa và giải mã

Mã hóa đối xứng

- Nhà cung cấp "SunJCE" đề nghị các giải thuật sau:
  - DES: mặc định khóa có chiều dài 56 bit
  - Triple DES: mặc định khóa có chiều 112 bit
  - Blowfish: mặc định khóa có chiều 56 bit
- Kích thước khóa có thể thay đổi tùy thuộc vào phiên bản của JCE

Khóa

- Hai phương pháp tạo khóa
  - Phương pháp đơn giản (cấp cao)
    - sinh khóa với lớp javax.crypto.KeyGenerator
  - Phương pháp nâng cao (cấp thấp)
    - Xây dựng khóa với các lớp:
    - javax.crypto.spec.algoKeySpec (DESKeySpec, BlowfishKeySpec)
    - java.security.KeyFactory

Sinh khóa

- Tạo ra và khởi động bộ sinh khóa

```
KeyGenerator gen;
gen = KeyGenerator.getInstance(“algo”);
gen.init();
```

- Sinh khóa dùng bộ sinh khóa

```
SecretKey key;
key = gen.generateKey();
```

Xây dựng khóa (pp 1)

- Chọn dữ liệu nhị phân dùng để tạo khóa: mảng các byte
- Xây dựng khóa sử dụng giải thuật và nhà cung cấp dịch vụ: DESKeySpec, DESedeKeySpec
- Xây dựng "nhà máy sản xuất khóa": một đối tượng của lớp KeyFactory
- Sinh khóa: generateSecret()