Эссе по курсу «Защита информации», кафедра радиотехники,
Московский Физико-Технический Институт (МФТИ ГУ)

Гонтарев Александр

Обзор Java Crypto API

Москва, 2007

1. Введение

Многие программисты в процессе своего профессионального роста рано или поздно сталкиваются с проблемой защиты своих программ от взлома. Он проявляется в самых разных местах готового программного продукта, начиная от попытки нахождения схем генерирования серийных номеров и заканчивая гораздо более серьезными атаками: попытками дешифрования конфиденциальных данных пользователя, хранящихся на всевозможных носителях или передающихся по различным сетям. Предположим, например, вы создали прекрасный клиент электронной почты, но в нем не хватает маленькой детали: письма на диске хранятся в открытом виде. Может быть, в этом ничего страшного и нет, но когда пользователю приходят в письмах важные данные, или он просто не желает, чтобы кто-то прочитал его почту, что остается делать программистам?

Часто они начинают сами придумывать схемы шифрования, хранения паролей и т.п. Хорошо, если программист знаком с наукой и действительно понимает то, что делает. Вообще говоря, разработка надежной системы безопасности программного обеспечения может занять много времени, которое могло бы пойти на улучшение функциональности, выявление и устранения других ошибок, которые, как ни крути, встречаются даже в дорогих продуктах от весьма серьезных производителей.

В последнее время большое число крупных программных продуктов реализуются на языке Java. Разработчик языка, компания Sun Microsystems, предоставляет встроенные возможности для шифрования данных и построения безопасных программ.

2. Структура Security API

Java Security API предоставляется в виде набора пакетов и классов, используемых для написания приложений. В их состав входят

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

Пакеты java.security и javax.crypto содержат классы, отображающие основные понятия криптографии.

java.security.cert.Certificate – сертификат.

javax.crypto.Cipher – шифр.

java.security.Key , java.security.PrivateKey , java.security.PublicKey , javax.crypto.SecretKey – ключи, используемые для шифрования и подписи.

javax.crypto.KeyAgreement – протокол передачи секретных ключей.

javax.crypto.KeyGenerator – производит ключи для симметричных шифров.

java.security.KeyPairGenerator – производит пары секретный ключ-секретный ключ для шифрования или цифровой подписи.

javax.crypto.Mac – Message Authentication Code (MAC)

java.security.MessageDigest – hash-функции

java.security.SecureRandom – генератор случайных чисел.

java.security.Signature – цифровая подпись.

Все методы классов делятся на две части – Application Programming Interface и Service Provider Interface (методы, которые надо реализовать самому).

3. Security API

3.1 Случайные числа.

Генерация случайного числа – наиболее важная функция во всей криптосистеме, потому что она пользуется для генерации ключей. Вся ответственность за ее реализацию ложится на разработчиков криптопровайдера. Начиная с версии JDK 1.1 существует класс java.security.SecureRandom – генератор псевдослучайных чисел. Если в явном виде не указать начальное состояние генератора, то начальное состояние генератора будет получена из информации, основанной на взаимодействии нитей исполнения в системе.

```
SecureRandom sr = new SecureRandom();
byte[] pseudoRandom = new byte[100];
sr.nextBytes(pseudoRandom);
```

3.2 Ключи

Интерфейс java.security.Key инкапсулирует ключ. Его расширяют несколько других интерфесов – java.security.PublicKey (открытый ключ), java.security.PrivateKey (используется вместе с предыдущим, секретный ключ).

Для JCE доступны так же javax.crypto.SecretKey – ключ, используемый в симметричных шифрах и java.security.KeyPair – пара секретного и публичного ключей.

Ключи создаются специальным классом, называемым key generator. Чтобы получить новый ключ, нужно

- получить генератор
- инициализировать его
- запросить у генератора ключ

java.security.KeyPairGenerator может сгенерировать пару открытый\закрытый ключ.

Получить его можно следующим образом:

```
KeyPairGenerator kpg = KeyPairGenerator.getInstance("ElGamal");
```

Если текущий криптопровайдер не знает об указанном методе, то будет выброшена исключение NoSuchAlgorithmException.

Далее надо инициализации вызывается метод

public void initialize(int strength)

Теперь можно запросить новый ключ:

```
public abstract KeyPair genKeyPair()
```

javax.crypto.KeyGenerator генерирует ключ для симметричного шифра.

Порядок действий такой же, как и прежде:

```
KeyGenerator kg = KeyGenerator.getInstance("DES");
kg.init(new SecureRandom());
SecretKey key = kg.generateKey();
```

3.3 Хэш-функции

Хэш-функции реализуются классами, наследующимися от java.security.MessageDigest. Для того, чтобы посчитать хэш от произвольного значения, нужно получить объект, реализующий функцию. Далее на вход ему задать данные для подсчета функции и посчитать значение:

```
// Define byte[] inputData first.
MessageDigest md = MessageDigest.getInstance("SHA");
md.update(inputData);
byte[] digest = md.digest();
```

3.4 MACs

Mac инкапсулирует javax.crypto.Mac. Для использования необходимо получить объект-реализацию алгоритма и инициализировать его некоторым ключом.

```
SecureRandom sr = new SecureRandom();
byte[] keyBytes = new byte[20];
sr.nextBytes(keyBytes);
SecretKey key = new SecretKeySpec(keyBytes, "HmacSHA1");
Mac m = Mac.getInstance("HmacSHA1");
m.init(key);
m.update(inputData);
byte[] mac = m.doFinal();
```

3.5 Цифровые подписи.

Цифровые подписи инкапсулируются классом java.security.Signature.
Как и во всех других случаях, необходимо получить экземпляр класса
public static Signature getInstance(String algorithm) throws NoSuchAlgorithmException
Далее надо инициализировать полученный экземпляр:
`public final void initSign(PrivateKey privateKey) throws InvalidKeyException` – для генерации подписи
`public final void initVerify(PublicKey publicKey) throws InvalidKeyException` – для проверки подписи
Добавляем данные для подписи
`public final void update(byte input) throws SignatureException`
И после этого можем генерировать и проверять подписи:
`public final byte[] sign() throws SignatureException` – генерация
`public final boolean verify(byte[] signature) throws SignatureException` – проверка подлинности подписи.

3.6 Сертификаты

Сертификат представляется в виде класса java.security.cert.Certificate. Одним из популярных алгоритмов является X.509. Его реализацию можно найти в классе java.security.cert.X509Certificate.
Сертификат загружается с помощью метода
`public static final X509Certificate getInstance(byte[] certData) throws CertificateException`
После этого можно получать данные из сертификата.

3.7 Симметричные и несимметричные шифры.

Для реализации возможности шифрования\расшифрования реализован класс javax.crypto.Cipher. Его используют в три этапа:

- Получить Cipher с помощью метода getInstance()

  При создании объекта, необходимо указать алгоритм и метод шифрования. На данный момент поддерживаются методы ECB , CBC , CFB , OFB и PCBC.
- Инициализировать его методом init(), все методы принимают значение Cipher.ENCRYPT_MODE или Cipher.DECRYPT_MODE для определения направления действия класса и ключ
- Зашифровать\расшифровать данные, используя update() и doFinal().

```
Cipher cipher = Cipher.getInstance("DES/ECB/PKCS5Padding");
cipher.init(Cipher.ENCRYPT_MODE, key);
byte[] raw = cipher.doFinal(stringBytes);
```

4. Заключение

Таким образом, программисту предоставлен весь спектр возможностей построения системы безопасности (от генерации случайных чисел, до блочных шифров).
На официальном сайте компании Sun представлено краткое описание провайдера SunJCE. В частности, там указаны все реализованные на данный момент алгоритмы

(http://java.sun.com/j2se/1.5.0/docs/guide/security/jce/JCERefGuide.html#AppA) – весьма внушительный список.
Кроме того, существуют другие криптопровайдеры – не только компании Sun. Часть из них доступна в свободном доступе. Например, провайдер Cryptix (http://www.cryptix.org/).

5. Использованная литература

- Java 2 Platform SE 5.0 API Specification (http://java.sun.com/j2se/1.5.0/docs/api/)
- Java cryptography by Jonathan Knudsen
- Cryptix project (http://www.cryptix.org/)