technische universität dortmund

Prof. Dr. Jan Jürjens, Thomas Ruhroth
Lehrstuhl 14 – Software Engineering
Fakultät für Informatik
SFL – 11.12.2012

Sicherheit: Fragen und Lösungsansätze – Übung 5

Hinweis: Wenn der Text länger als vorgegeben ist, wird der Rest ignoriert.
Abgabe: Ausnahmsweise dieses eine Mal als Email an mich.
(Adresse unter http://www-jj.cs.tu-dortmund.de/staff/ruhroth). Abgabe spätestens am 21. Dezember.

AUFGABE 1 (One Way Hash) (45 Min, 5p):

- Implementieren Sie folgende einfache Hash-Funktion als einfache Javaoperation `public String hash(String in)`:

  Es sei folgende Hash-Funktion gegeben: Für einen Text wird die Anzahl der einzelnen Buchstaben (Groß/Kleinschreibung wird ignoriert) bestimmt und als Ziffernfolge aller Buchstabenzahlen ohne führende Nullen im Dezimalsystem ausgegeben. Z.B. führt der Text "Implementieren Sie folgende einfache Hash-Funktion:" zu Hashwert "20119312501226210122100000".

- Ist die implementierte Hash-Funktion eine One-Way-Hash-Funktion? Wenn ja, begründen Sie Ihre Antwort (Max. 50 Worte). Wenn Nein, Implementieren Sie die Umkehrfunktion.

- Ist die Hash-Funktion kollisionsfrei bzw. schwach kollisionsfrei? Betrachten Sie als Domäne alle möglichen Strings, alle deutschen Texte und die Menge aller Shakespeare Sonette (Max. 100 Worte).

AUFGABE 2 (Kryptographie mit Java: Implementierung einer Hashfunktion) (45 Min, 5p):

Betrachten Sie die (einfache und für die Praxis ungeeignete!) Hashfunktion h, die einen String auf seinen Anfangs- und Endbuchstaben abbildet, z. B.:

$$h(Alice) = Ae, h(Bob) = Bb, h(Hallo!) = H!usw.$$

Implementieren Sie diese Hashfunktion h nun im Rahmen der Java Cryptography Architecture und stellen Sie sie mittels eines Providers zur Verwendung bereit. Nehmen Sie dazu die Java- Schnittstellen-Dokumentation zu Hilfe (http://download.oracle.com/javase/6/docs/api/) und gehen Sie wie folgt vor:

1. Das Service Provider Interface für Hashfunktionen ist `MessageDigestSpi`. Es handelt sich dabei um eine abstrakte Klasse, deren abstrakte Methoden Sie in einer abgeleiteten Klasse implementieren müssen.

(a) Erstellen Sie eine als final deklarierte und von `MessageDigestSpi` abgeleitete Klasse `SimpleHash`.

(b) Die Klasse SimpleHash soll ein internes Byte-Array unterhalten, in dem die Byte-Darstellung des Strings gespeichert wird, dessen Hashwert berechnet werden soll. Implementieren Sie dazu die folgenden in MessageDigestSpi abstrakten Methoden:

- void engineReset(): löscht den Inhalt des internen Byte-Arrays.
- void engineUpdate(byte input): hängt input an das interne Byte-Array an.
- void engineUpdate(byte[] input, int offset, int len): hängt den Teil von input an das interne Byte-Array an, der bei offset beginnt und eine Länge von `len` hat.
- byte[] engineDigest(): berechnet den Hashwert des internen Byte-Arrays und gibt ihn in Form eines Byte-Arrays zurück.

2. Implementieren Sie den als final deklarierten Provider `SimpleHashProvider`. Orientieren Sie sich an der folgenden Anleitung (How to Implement a Provider in the JavaTM Cryptography Architecture): http://download.oracle.com/javase/6/docs/technotes/guides/security/crypto/HowToImplAProvider.html#Step3

3. Implementieren Sie schließlich wie folgt eine Testklasse SimpleHashProviderTest:

   (a) Zuerst muss der SimpleHashProvider mit der addProvider-Methode aus der Klasse `java.security.Security` dynamisch registriert werden.

   (b) Geben Sie nun den Namen (`getName()`), die Version (`getVersion()`) sowie die Beschreibung (`getInfo()`) des Providers aus.

   (c) Erzeugen Sie dann mit der getInstance-Methode der engine class `MessageDigest` ein Objekt, das Ihre Hashfunktion ausführen kann. Lassen Sie dieses Objekt mit der getAlgorithm-Methode den Algorithmus ausgeben, den es verwendet.

   (d) Verifizieren Sie wie folgt, ob Sie die Hashfunktion korrekt implementiert haben:

   i. Lesen Sie mit der Methode update(byte[] input) den String foo ein, berechnen Sie mit digest() seinen Hashwert und geben Sie diesen Hashwert aus.

   ii. Hängen Sie mittels update(byte[] input) den String bar an den bereits gespeicherten String an, berechnen Sie den Hashwert und geben Sie ihn aus.

   iii. Löschen Sie mit der Methode reset() den gespeicherten String. Lesen Sie dann mit der Methode update(byte[] input, int offset, int len) von dem String hello world die Positionen 2-8 ein, berechnen Sie den Hashwert und geben Sie ihn aus.

Hinweis: Um einen String in ein Byte-Array umzuwandeln, benutzen Sie die getBytes-Methode. Für die Rück-Umwandlung benutzen Sie den entsprechenden String-Konstruktor:

```
String str1 = "foo";          // ein String,...
byte[] ba = str1.getBytes();  // ...die Umwandlung in ein Byte-Array...
String str2 = new String(ba); // ...und die Rück-Umwandlung in einen String
```