Programmierpraktikum
Thema: Kryptographie
Sommersemester 2004

3. Übungsblatt

1. (*) Es wurde folgender Chiffretext abgefangen:

```
uqu jxoheq, rnve rvhes adfwbq ewpk pof cusy vqhmhz
gboxhr, eqjbu pwrvqg wdtfmhtbg cg sagq urkf, svqqb
zhzgpkqb nxr rrp ycag li ydzrrq gbq ltb flovru iwrgqf
mxd segq nhugspnliouubthz. yrlz oagqfrv bfbmqyg gqf
edgasdtfg lz iavqfru lsvw iweg riru pwr pqbfftvrlf
svqpfhfwgirxzru gbq igse gus ydzusuugglss rurcevovhqs
rrv isywdohpe prgqighzrru gbq nqwahe kvup gpkiwruuuru
li ruriroxsa vqwa... dnse lz svqqa glqtruqb flzb jldr
rv mipk zwpkf bhu qwa pmba vqwa gqf mxy abqp ohinfvfth,
frzrruz krqz kvu zie hzhfftzbvesa vubq, gus tdznr
qmhvrz.
```

   Es ist bekannt, dass das Vigenère-Verfahren benutzt wurde. Bestimme zuerst die Schlüssellänge mit Hilfe des Kasiski- oder Friedman-Tests. Wie lautet dann das Schlüsselwort und der zugehörige Klartext?

   Hinweis 1: Leerzeichen, Satzzeichen sowie Zeilenumbruch wurden wie üblich nicht mitverschlüsselt!

   Hinweis 2: Der Klartext steht in elektronischer Form auf der Webseite zur Verfügung.

2. (Alexander/Oleg/Yves) Der Friedman-Test soll mit der Funktion

```
double friedman()
```

   in der Klasse `Analyze` realisiert werden. Es wird kein String übergeben, da auf der Variablen `ciphertext` gearbeitet wird. Man beachte, dass nur Kleinbuchstaben (ASCII-Werte 97-122) für die Analyse betrachtet werden. Der Rückgabewert der Funktion soll dem Erwartungswert für die Schlüssellänge entsprechen:

$$h = \frac{0.0377n}{I * (n-1) - 0.0385n + 0.0762}$$

   n ist die Länge des Textes (Erinnerung: nur Kleinbuchstaben zählen!), I ist der Friedmansche Koinzidenzindex (n_i Häufigkeit des $i-ten$ Buchstabens im Text):

$$I = \frac{\sum_{i=1}^{26} n_i(n_i - 1)}{n(n-1)}$$

3. (Oliver/Philipp/Tim) Die Klasse `CryptFile` muß um Methoden für das Lesen und Schreiben von Challenges und Solutions erweitert werden. Diese werden durch die Klassen `Challenge` und `Solution` im package `crypto` repräsentiert. Folgende Funktionen werden benötigt:

   ```
   Solution readSolution(int buddyID)
   boolean writeSolution(int buddyID, Solution s)
   Challenge readInChallenge(int buddyID)
   boolean writeInChallenge(int buddyID, Challenge c)
   Challenge readOutChallenge(int buddyID)
   boolean writeOutChallenge(int buddyID, Challenge c)
   boolean compareSolution(int buddyID, Solution s)
   ```

   Es kann pro Buddy nur jeweils eine InChallenge und eine OutChallenge gespeichert sein, deswegen wird keine Challenge ID verlangt. Die InChallenge wurde vom Kommunikationspartner generiert und soll mit den bekannten Methoden dechiffriert werden. Die OutChallenge wurde zum Dekodieren an den Partner gesendet.

   Sobald dieser eine Solution zurücksendet, wird diese mit der gespeicherten OutChallenge verglichen. Die Methode `compareSolution` vergleicht die erhaltene Solution mit der gespeicherten OutChallenge und liefert den Wert `true` zurück, falls die Variablen `String text` übereinstimmen.

   Für die Challenges legt man zwei Dateien an, `Challenge<BuddyID>.in` und `Challenge<BuddyID>.out`. Die Datei `Solution.<BuddyID>` wird für die Solutions angelegt. Die Methode `getXML()` in den jeweiligen Klassen schreibt vor, wie das XML in den gespeicherten Dateien auszusehen hat.

4. (David/Pavel) Für die drei ausgewählten Verfahren zum Entschlüsseln werden drei verschiedene GUIs für die Kryptanalyse benötigt. Man soll sich an den Vorlagen im Skript (Teil 2) orientieren.

Abgabe: bis Dienstag, 15.6.2004