Angewandte Kryptographie — Praktikum 1

Thema: Vigenère Chiffre — Wintersemester 2025/2026

Prof. Dr. Christoph Karg — Hochschule Aalen

Ziel dieses Praktikums ist die Implementierung der Vigenère Chiffre und der zugehörigen Analysemethoden.

Aufgabe 1.

a) Entpacken Sie die auf der Homepage der Vorlesung bereit gestellte TGZ-Datei im Wurzelverzeichnis des Projekts. Auf diese Weise werden die Dateien an den korrekten Stellen eingefügt.

b) Verschaffen Sie sich einen Überblick über die im Archiv enthaltenen Dateien.

c) Fügen Sie in die im Verzeichnis `AK/src`[1] befindliche `CMakeLists.txt` Datei[2] die Zeile

```
ADD_SUBDIRECTORY( Praktikum-Vigenere )
```

ein und aktualisieren Sie Ihren Projekt Build. Wenn beim Übersetzen des Projekts an dieser Stelle Fehler auftreten, dann ist dies nicht weiter schlimm, da die entsprechenden Quelldateien noch nicht komplett implementiert sind.

Aufgabe 2. Implementieren Sie die Methoden `encrypt()` und `decrypt()` der Klasse `VigenereCipher`. Testen Sie Ihre Implementierung unter Verwendung des Konsolenprogramms `vigenere`. Die Kommandozeilenparameter des Programms befinden sich in der Datei `vigenere-getopt.ggo`.

Aufgabe 3.

a) Verschlüsseln Sie mit der Vigenère Chiffre den Klartext

```
angewandtekryptographie
```

mit dem Schlüssel `blau`.

b) Entschlüsseln Sie mit der Vigenère Chiffre den Klartext

```
RBZVKTERMVYKPDMFUKRDAZS
```

mit dem Schlüssel `rot`.

---

[1] Es wird angenommen, dass das Wurzelverzeichnis des Projekts `AK` heißt. Dieser Name muss gegebenfalls angepasst werden.

[2] Diese Datei stammt aus dem Praktikum zu CMake.

Aufgabe 4. Die Methode `kasiskiTest()` der Klasse `VigenereBreaker` dient zur Durchführung des Kasiski Tests auf einen gegebenen Geheimtext. Die Methode hat die folgenden Parameter:

- `cipher_text` $\rightsquigarrow$ zu analysierender Geheimtext
- `ngram_len` $\rightsquigarrow$ Länge der zu bearbeitenden n-gramme
- `verbose` $\rightsquigarrow$ Aktivierung zusätzlicher Ausgaben (Default: `false`)

Die Methode soll den am häufigsten vorkommenden Wert des größten gemeinsamen Teilers zurückgeben.

Implementieren Sie diese Methode, indem Sie in folgenden Schritten vorgehen:

a) Konvertieren den Inhalt von `cipher_text`, in einen String vom Typ `string`, der den Geheimtext in Großbuchstaben enthält.

b) Speichern Sie die im Text vorkommenden n-gramme und die zugehörigen Abstände zum Textanfang in einer Tabelle vom Typ `map<string, vector<int> >`.

   Hinweis: Die Klasse `string` bietet unter anderem die Methode `substr(p,n)`, die das Teilwort zurückliefert, welches im String an Position `p` beginnt und die Länge `n` hat. Der Rückgabewert ist ein `string`-Objekt.

c) Berechnen Sie für jedes n-gramm, das mindestens dreimal vorkommt, den größten gemeinsamen Teiler der relativen Abstände zur Position des ersten Vorkommens.

   Hinweis: Benutzen Sie zur Berechnung des größten gemeinsamen Teilers die in der Klasse `VigenereBreaker` bereitgestellten `gcd`-Methoden.

d) Geben Sie für jedes n-gramm mit mindestens drei Vorkommen die entsprechenden Positionen im Geheimtext sowie den zugehörigen größten gemeinsamen Teiler aus.

Benutzen Sie das Konsolenprogramm `kasiski`, um Ihre Methode aufzurufen. Die Kommandozeilenparameter des Programms befinden sich in der Datei `kasiski-getopt.ggo`.

Aufgabe 5. Führen Sie auf den Inhalt der Datei `vc-praktikum-cipher.txt` einen Kasiski Test durch und ermitteln Sie die Schlüssellänge des mit der Vigenère Chiffre kryptierten Textes.

Aufgabe 6. Implementieren Sie die Methode `coincidenceIndex` der Klasse `VigenereBreaker` zur Berechnung des Koinzidenzindex eines gegebenen Textes. Die Methode besitzt folgenden Parameter:

- `text` $\rightsquigarrow$ zu analysierender Text

Die Methode gibt den Koinzidenzindex des Textes als `float` zurück.

Aufgabe 7. Implementieren Sie die Methode `coincidenceTest` der Klasse `VigenereBreaker` zur Durchführung des Koinzidenzmethode zur Bestimmung der Schlüssellänge. Die Methode besitzt folgende Parameter:

- `cipher_text` ⇝ zu analysierender Geheimtext
- `cols` ⇝ Spaltenanzahl (= Schlüssellänge)
- `threshold` ⇝ Schwellwert (Default: 0.065)
- `verbose` ⇝ Aktivierung zusätzlicher Ausgaben (Default: `false`)

Die Methode soll genau dann `true` zurückliefern, wenn die Indizes aller Spalten über dem Schwellwert `threshold` liegen. Die Koinzidenzindizes der einzelnen Spalten sollen in einer Zeile ausgeben werden.

Setzen Sie das Konsolenprogramm `coindex` ein, um auf die obige Methode zuzugreifen. Die Kommandozeilenparameter des Programms befinden sich in der Datei `coindex-getopt.ggo`.

Aufgabe 8. Analysieren Sie den Inhalt der Datei `vc-praktikum-cipher.txt` und ermitteln Sie mittels der Koinzidenzindex-Methode die Schlüssellänge.

Aufgabe 9. Implementieren Sie die Methode `mutualCoinIndex` der Klasse `Vigenere Breaker` zur Berechnung des gegenseitigen Koinzidenzindex zweier Geheimtextspalten. Die Methode hat folgende Parameter:

- `cipher_text` ⇝ zu analysierender Geheimtext
- `cols` ⇝ Spaltenanzahl
- `col_i` ⇝ erste Spalte
- `col_j` ⇝ zweite Spalte
- `threshold` ⇝ Schwellwert (Default: 0.065)
- `verbose` ⇝ Aktivierung zusätzlicher Ausgaben (Default: `false`)

Die Methode soll die Verschiebung mit maximalem Index zurückgeben, falls dieser Index größer-gleich `threshold` ist. Andernfalls soll -1 zurückgegeben werden.

Verwenden Sie das Konsolenprogramm `searchshift`, um die Implementierung Ihrer Methode zu testen. Die Kommandozeilenparameter des Programms befinden sich in der Datei `searchshift-getopt.ggo`.

Hinweis: Geben Sie für jede Verschiebung den berechneten gegenseitigen Koinzidenzindex auf dem Bildschirm aus.

Aufgabe 10. Entschlüsseln Sie die Datei `vc-praktikum-cipher.txt`, indem Sie den eingesetzten Schlüssel finden und mit diesem die Datei entschlüsseln.