

## Lastenheft „Secure Text Editor“

Ziel ist die Implementierung eines *Secure Text Editors* (STE), der Textdateien unter Verwendung verschiedener kryptografischer Verfahren schützen kann.

### Funktionale Anforderungen

Der STE MUSS über grundsätzliche Funktionen eines Editors verfügen und MUSS das Erstellen und Bearbeiten von Texten ermöglichen. Die Texte MÜSSEN beim Abspeichern und Laden verschlüsselt bzw. entschlüsselt werden können. Folgende kryptografische Verfahren MÜSSEN zur Auswahl stehen:

- Symmetrische Verschlüsselung:
  - AES mit Schlüssellängen 128, 192, 256 Bit
- Passwort-basierte Verschlüsselung:
  - AES 256 Bit, GCM, SCRYPT
  - PBESWithSHA256And128BitAES-CBC-BC
  - PBESWithSHAAnd40BitRC4
- Digitale Signatur (Erzeugung und Verifikation):
  - SHA256withDSA

In Abhängigkeit des gewählten Verfahrens, MÜSSEN die Padding Mechanismen `NoPadding`, `PKCS7Padding` und `ZeroBytePadding`, sowie als Blockmodi `ECB`, `CBC`, `OFB`, `CTS` und `GCM` zur Verfügung stehen. Der STE MUSS die benötigten Schlüssel generieren können.

Der STE MUSS Manipulationen an abgespeicherten Texten erkennen können, indem ein Hashwert mit abgespeichert und beim Laden des Textes verifiziert wird. Dazu MUSS der Nutzer auf folgende Hashfunktionen zurückgreifen können:

- SHA-256
- AESCMAC
- HMACSHA256

Das Speichern der Schlüssel, Texte und Hashwerte mit entsprechenden Metadaten MUSS (ggf. jeweils separat) in einem strukturierten Dateiformat (z. B. basierend auf XML) erfolgen.

### Nicht-funktionale Anforderungen

Der STE SOLL in Java und MUSS mithilfe der aktuellsten Version von `The Legion of the Bouncy Castle`<sup>1</sup> implementiert werden. Die Nutzerinteraktion SOLL über eine grafische Benutzeroberfläche erfolgen. Die Softwarearchitektur MUSS eine einfache Erweiterbarkeit erlauben, sodass zusätzliche Kryptografiekomponenten leicht ergänzt werden können.

<sup>1</sup><https://www.bouncycastle.org>, aufgerufen am 31.01.2019

Der STE MUSS inline dokumentiert werden. Dazu MUSS für Java Javadoc und Doxygen<sup>2</sup> verwendet werden. Weiterhin MUSS eine Validierung basierend auf dem Unit-Test-Framework implementiert werden Dazu MUSS für Java JUnit<sup>3</sup> verwendet werden. Weiterhin MUSS eine Analyse der Testabdeckung hinsichtlich Anweisungs-, Zweig- und Bedingungsabdeckung mithilfe eines Werkzeugs wie z. B. JaCoCo<sup>4</sup> durchgeführt werden.

Idealerweise arbeiten Sie mit IntelliJ IDEA<sup>5</sup> oder Eclipse<sup>6</sup>, Git<sup>7</sup> und Maven<sup>8</sup>.

IntelliJ IDEA Ultimate ist als Teil des JetBrains Product Pack for Students kostenlos for educational use only durch Registrierung auf <https://account.jetbrains.com/login> erhältlich.

---

<sup>2</sup><http://www.stack.nl/~dimitri/doxygen/>, aufgerufen am 31.01.2019

<sup>3</sup><http://junit.org/>, aufgerufen am 31.01.2019

<sup>4</sup><http://www.eclemma.org/jacoco/>, aufgerufen am 31.01.2019

<sup>5</sup><https://www.jetbrains.com/idea/>, aufgerufen am 31.01.2019

<sup>6</sup><https://eclipse.org/>, aufgerufen am 31.01.2019

<sup>7</sup><https://git-scm.com/>, aufgerufen am 31.01.2019

<sup>8</sup><https://maven.apache.org/>, aufgerufen am 31.01.2019