

**Projektarbeit (Informatik)**

Sichere eTest auf eBook Readern

|  |  |  |
| --- | --- | --- |
| **Autoren** |  | Simon Lukes  Daniel Jampen |
| **Hauptbetreuung** |  | Karl Rege |
| **Datum** |  | 11.12.2014 |



Zusammenfassung

Die Papierform ist immer noch die aktuelle Wahl, wenn ein sicherer und kontrollierbarer Prüfungsablauf gefordert ist. Doch je länger je mehr wird versucht, diese mittels Neuentwicklungen auf elektronischer Basis abzulösen, welche jedoch bis anhin kein zufriedenstellendes Resultat lieferten. Teile der Grundproblematiken von eTests waren bei allen Versuchen auszumachen: Manipulationen, unzulässige Infrastruktur oder prüfungsinterne Kommunikation beeinträchtigen die Verlässlichkeit der Prüfungsresultate. In dieser Arbeit wird demonstriert, wie die Grundproblematiken von eTests aktiv angegangen und verhindert werden können. Die entwickelte Applikation SecureExam bietet eine Plattform zum Erstellen und Durchführen von sicheren und zuverlässigen ePrüfungen. SecureExam generiert aus bestehenden Fragen und einer Teilnehmerliste eine AES verschlüsselte Prüfungsdatei in HTML-Format, welche plattformunabhängig für Prüfungsszenarien eingesetzt werden kann. Zur Authentifizierung werden aus den Teilnehmerdaten, pro Absolvent, individuelle Zugangsdaten generiert, welche zu Beginn der Prüfung zur Sicherstellung der Identität des Probanden eingegeben werden müssen. Mit dem Starten der Prüfung werden verschiedene, transparente Manipulationsmechanismen aktiv, welche alle Aktionen während der Gesamtdauer des Examens zur Nachverfolgung aufzeichnen und Betrugsversuche detektieren. Durch das dezentrale System von SecureExam wird der Infrastruktureinfluss minimalisiert und damit die Problematik der Unzuverlässigkeit von IT-Systemen abgeschwächt. Da das erarbeitete System auf keine externen Datenquellen angewiesen ist, kann durch geschickte Gerätewahl die Kommunikation während der Prüfung komplett unterbunden werden. Betrugsversuche können mit den gewählten Mitteln nicht direkt verhindert, aber von SecureExam zuverlässig entdeckt und festgehalten werden. Durch die Kombination der in dieser Arbeit entwickelten Ansätze kann ein sicherer, reibungsloser und kontrollierbarer Ablauf der Prüfung garantiert werden.

Abstract… english

Inhaltsverzeichnis

[1 Einleitung 9](#_Toc406022897)

[1.1 Ausgangslage 9](#_Toc406022898)

[1.1.1 Mögliche Probleme 9](#_Toc406022899)

[1.2 Aufgabenstellung 10](#_Toc406022900)

[1.3 Anforderungen 10](#_Toc406022901)

[1.3.1 Muss Funktionen 10](#_Toc406022902)

[1.3.2 Wunsch Funktionen 10](#_Toc406022903)

[1.4 Zielsetzung 11](#_Toc406022904)

[1.5 Zielpublikum 11](#_Toc406022905)

[2 Grundlagen 12](#_Toc406022906)

[2.1 Kryptographie 12](#_Toc406022907)

[2.1.1 Verschlüsselung 12](#_Toc406022908)

[2.1.2 Passwort Hashing 13](#_Toc406022909)

[2.2 Dateiformate 14](#_Toc406022910)

[2.2.1 INI 14](#_Toc406022911)

[2.2.2 XML 14](#_Toc406022912)

[2.2.3 Open Office 15](#_Toc406022913)

[2.2.4 Microsoft Word 15](#_Toc406022914)

[2.2.5 XSLT 15](#_Toc406022915)

[2.2.6 PDF 16](#_Toc406022916)

[2.2.7 HTML mit Javascript 16](#_Toc406022917)

[2.3 Digitale Manipulationsmöglichkeiten 16](#_Toc406022918)

[2.3.1 Korrekte Antwort auslesen 16](#_Toc406022919)

[2.3.2 Zeit 16](#_Toc406022920)

[2.3.3 Internet Zugriff 16](#_Toc406022921)

[2.3.4 Prüfungsinterne Kommunikation 17](#_Toc406022922)

[2.4 eBook Reader 17](#_Toc406022923)

[2.4.1 Einschränkungen 17](#_Toc406022924)

[3 Analyse 19](#_Toc406022925)

[3.1 Anwendungsfälle 19](#_Toc406022926)

[3.1.1 Übersicht 19](#_Toc406022927)

[3.1.2 Akteure 19](#_Toc406022928)

[3.1.3 AF01: Prüfung erstellen 19](#_Toc406022929)

[3.1.4 AF02 Prüfung absolvieren 21](#_Toc406022930)

[3.2 Nichtfunktionale Anforderungen 22](#_Toc406022931)

[3.2.1 NFA01 Prüfung ist einfach bedienbar 22](#_Toc406022932)

[3.2.2 NFA02 Export erfolgt transparent für den Probanden 22](#_Toc406022933)

[3.2.3 NFA03 Anmeldevorgang dauert weniger als drei Sekunden 22](#_Toc406022934)

[4 Konzept 23](#_Toc406022935)

[4.1 Systemüberblick 23](#_Toc406022936)

[4.1.1 Domänenmodell 24](#_Toc406022937)

[4.2 C# Errorhandling 24](#_Toc406022938)

[4.3 Codedokumentation 25](#_Toc406022939)

[4.3.1 C# 25](#_Toc406022940)

[4.3.2 JavaScript 25](#_Toc406022941)

[4.4 Kryptographie 25](#_Toc406022942)

[4.4.1 Verschlüsselung 25](#_Toc406022943)

[4.4.2 Passwort hashing 27](#_Toc406022944)

[4.5 Konfigurationsdateien 28](#_Toc406022945)

[4.5.1 Entscheid 28](#_Toc406022946)

[4.5.2 Konzept 28](#_Toc406022947)

[4.6 Import 29](#_Toc406022948)

[4.6.1 Evaluationsmatrix 29](#_Toc406022949)

[4.6.2 Entscheid 29](#_Toc406022950)

[4.6.3 Konzept 29](#_Toc406022951)

[4.7 Export 29](#_Toc406022952)

[4.7.1 Evaluationsmatrix 29](#_Toc406022953)

[4.7.2 Entscheid 30](#_Toc406022954)

[4.7.3 Konzept 30](#_Toc406022955)

[4.8 Digitale Manipulationsmöglichkeiten 30](#_Toc406022956)

[4.8.1 Zeit 30](#_Toc406022957)

[4.8.2 Internet Zugriff 31](#_Toc406022958)

[4.8.3 Prüfungsinterne Kommunikation 31](#_Toc406022959)

[4.9 UI-Design 31](#_Toc406022960)

[4.9.1 Idee 31](#_Toc406022961)

[4.9.2 Login 32](#_Toc406022962)

[4.9.3 Prüfung 33](#_Toc406022963)

[4.10 JavaScript 34](#_Toc406022964)

[4.10.1 SecureExam Library 34](#_Toc406022965)

[4.10.2 CryptoJS Library 34](#_Toc406022966)

[4.10.3 FileSaver.js Library 35](#_Toc406022967)

[4.10.4 Code Obfuscation 35](#_Toc406022968)

[5 Umsetzung 36](#_Toc406022969)

[5.1 Kryptographie 36](#_Toc406022970)

[5.1.1 Verschlüsselung 36](#_Toc406022971)

[5.1.2 Entschlüsselung 36](#_Toc406022972)

[5.1.3 Passwort hashing 37](#_Toc406022973)

[5.2 Konfigurationsdateien 38](#_Toc406022974)

[5.2.1 Prüfungskonfigurationsdatei 38](#_Toc406022975)

[5.2.2 SecureExam.xml 39](#_Toc406022976)

[5.3 Import 39](#_Toc406022977)

[5.3.1 XML 40](#_Toc406022978)

[5.3.2 Open Office 40](#_Toc406022979)

[5.3.3 Studenten Import 41](#_Toc406022980)

[5.4 Export 41](#_Toc406022981)

[5.4.1 Prüfungsdatei 41](#_Toc406022982)

[5.4.2 Studenten Login Daten 42](#_Toc406022983)

[5.5 Digitale Manipulationsmöglichkeiten 42](#_Toc406022984)

[5.5.1 Zeit 42](#_Toc406022985)

[5.5.2 Internet Zugriff 45](#_Toc406022986)

[5.6 UI-Design 45](#_Toc406022987)

[5.6.1 Farben 45](#_Toc406022988)

[5.6.2 Authentifizierungsansicht 45](#_Toc406022989)

[5.6.3 Page-Mode 46](#_Toc406022990)

[5.6.4 Scroll-Mode 46](#_Toc406022991)

[6 Testing 47](#_Toc406022992)

[6.1 Konzept 47](#_Toc406022993)

[6.2 Realisierung 47](#_Toc406022994)

[6.2.1 Fakes 47](#_Toc406022995)

[6.2.2 Konsolenausgabe abfangen 48](#_Toc406022996)

[6.3 Testresultat 48](#_Toc406022997)

[6.3.1 Unit Tests 48](#_Toc406022998)

[6.3.2 Code Coverage Analyzer 49](#_Toc406022999)

[7 Resultate 49](#_Toc406023000)

[7.1 Scrollbare Prüfung auf PC 49](#_Toc406023001)

[7.2 Prüfung auf Tablet 49](#_Toc406023002)

[7.2.1 Tablet mit LCD-Bildschirm 49](#_Toc406023003)

[7.2.2 Tablet mit eInk Bildschirm 49](#_Toc406023004)

[8 Diskussion und Ausblick 49](#_Toc406023005)

[8.1 Erreichte Ergebnisse 49](#_Toc406023006)

[8.2 Ausblick und Chancen 49](#_Toc406023007)

[9 Verzeichnisse 50](#_Toc406023008)

[9.1 Literaturverzeichnis 50](#_Toc406023009)

[9.2 Glossar 50](#_Toc406023010)

[9.3 Abbildungsverzeichnis 50](#_Toc406023011)

[9.4 Tabellenverzeichnis 51](#_Toc406023012)

[10 Anhang 52](#_Toc406023013)

[10.1 Projektmanagement 52](#_Toc406023014)

[10.1.1 Zeitplan 52](#_Toc406023015)

[10.1.2 Sitzungsprotokolle 53](#_Toc406023016)

[10.1.3 C# Klassendiagramm 64](#_Toc406023017)

[10.1.4 SecureExam JavaScript Library 65](#_Toc406023018)

[10.2 Bedienungsanleitung 67](#_Toc406023019)

[10.2.1 Konsolenapplikation 67](#_Toc406023020)

[10.2.2 Generierte Prüfung 72](#_Toc406023021)

[10.3 Weiteres 72](#_Toc406023022)

[10.3.1 CD mit dem vollständigen Bericht als pdf-File und dem SourceCode 72](#_Toc406023023)

# Einleitung

Heutzutage werden Prüfungen fast ausschliesslich in Papierform durchgeführt. Dies obwohl die technischen Methoden zur Digitalisierung von Prüfungen schon lange vorhanden wären. Nicht nur würde eine Umstellung auf ein digitales Format die Korrekturarbeit der Professoren entlasten, es gibt auch diverse Vorteile für die Prüfungsabsolventen. In digitaler Form gibt es keine halb angekreuzten Checkboxen mehr und Korrekturen an falsch geschriebenen Wörtern/Antworten können während der Prüfung von den Studenten sauber gemacht werden. Weiter könnten zum Beispiel Programmierprüfungen in gewohntem Umfeld (über die Tastatur geschriebener Sourcecode) abgehalten werden. Noch viele weitere solcher Verbesserungsmöglichkeiten währen möglich und wären sehr einfach mit digitalen Prüfungen realisierbar.

Jedoch leider gibt es auch ein paar Schwachpunkte bzw. Herausforderungen, welche die digitalen Prüfungen bestehen müssten. Dadurch, dass man digital arbeitet, müssen auch die Manipulationsschutzmechanismen überarbeitet werden. Es muss sichergestellt werden, dass die Prüfungen sicher und ohne Bedenken abgehalten werden können.

## Ausgangslage

Fast alle Prüfungen werden bisher auf Papier gelöst, was unangenehme Probleme mit sich bringen kann. Jede Prüfung muss für jeden einzelnen Prüfungsteilnehmer ausgedruckt werden, was sowohl Druckkosten verursacht als auch aufwändig ist. Des Weiteren muss der Dozent alle Exemplare im Klassenzimmer austeilen, was je nach Raum unterschiedlich schnell erfolgt.

Eine Lösung zu diesen Problemen ist die digital Prüfung, welche allerdings auch nicht ganz ausgereift ist. Diverse Probleme wurden diesen Prüfungen bisher zum Verhängnis. Nachfolgend werden die Grundproblematiken aufgeführt und näher beschrieben:

### Grundproblematiken eTests

**Manipulationen:**

In der digitalen Welt gibt es keine absolute Sicherheit. Entsprechend sind Manipulationen in verschiedenen Formen generell immer möglich.

**Prüfungsdateien kommen abhanden:**

Da die Prüfung digital abgespeichert werden muss, besteht die Möglichkeit, dass Personen, welche Interesse an den Daten haben, versuchen diese Prüfungsdatei zu stehlen. Dies kann auf unterschiedlichste Art und Weise angegangen werden, zum Beispiel durch aktives Hacking oder durch die Unachtsamkeit eines Dozenten.

**Unzuverlässige Infrastruktur:**

Prüfungen beginnen oft an einem festgelegten Termin, entsprechend wird um diesen Zeitpunkt eine enorme Last auf die Prüfungsserver fallen, was oft zu Überlastungen führt. Diverse Einflüsse können die IT-Systeme zudem stören oder beeinträchtigen, was wiederum zu Ausfällen der ganzen Serverinfrastruktur führen kann.

**Gerätemissbrauch zur Kommunikation:**

Auch wenn ein in sich abgeschlossenes Netzwerk verwendet wird, sind die Geräte immer noch kommunikationsfähig. Durch geschickten Einsatz von Bluetooth oder W-LAN Ad-Hoc Netzwerken ist eine Kommunikation zwischen den Absolventen möglich.

## Aufgabenstellung

Prüfungen werden heute immer noch fast ausschliesslich auf Papier durchgeführt. Erste Versuche mit Online e-Tests mussten zum Teil abgebrochen oder wiederholt werden, weil entweder technische Probleme aufgetaucht sind oder die Tests manipuliert wurden. Es soll sich ein Verfahren ausgedacht und implementiert werden (Ideen bestehen bereits) wie e-Tests auf handelsüblichen eBook Readern sicher durchgeführt werden können. Dabei kommen die neusten HTML5 Technologien oder PDFs mit JavaScript zur Anwendung.

## Anforderungen

### Muss Funktionen

|  |  |
| --- | --- |
| Funktion | **Beschreibung** |
| Prüfungsgenerator erstellen | Es muss ein kommandozeilenbasiertes Programm entwickelt werden, welches aus einer Datei die Prüfungsfragen ausliest und aus diesen ein Formular im HTML-Format erstellt. |
| Prüfung ist eine Datei | Die Prüfung, welche mittels dem selber geschriebenen Programm generiert wird, besteht genau aus einer Datei. |
| Prüfung läuft auf eBook Reader | Die Prüfung muss auf einem eBook Reader durchführbar sein. Die kleinere Performance muss berücksichtigt werden. |
| Fragen verschlüsseln | Die Prüfungsfragen müssen in der Prüfungsdatei verschlüsselt sein. |
| Verschlüsselungsstärke | Jemand der die Prüfungsdatei stiehlt, darf nicht Fähig sein, die Fragen mit einem Aufwand von weniger als einem Tag entschlüsseln zu können. |
| Fragen live entschlüsseln | Nach der Authentifizierung des Prüfungskandidaten, werden die Fragen mittels einem, pro Student individuellen, Key entschlüsselt werden. |
| Keine Änderungen nach der Abgabe | Nachdem der Student die Prüfung mittels Klick auf „Abgeben“ beendet, darf dieser nicht mehr in der Lage sein, die Prüfung zu manipulieren oder weitere Fragen zu beantworten. |
| Zeitmanipulationsschutz | Die Prüfung merkt, wenn Manipulationen an der Systemuhr vorgenommen werden. |

### Wunsch Funktionen

|  |  |
| --- | --- |
| Funktion | **Beschreibung** |
| Internetverbindung detektieren | Die Prüfung besitzt die Funktionalität zu prüfen, ob das Gerät eine aktive Internetverbindung besitzt. |
| Import von Fragen aus Textverarbeitungsprogramm | Die Prüfungsfragen sollen aus einer, im Format definierten, ODT Datei extrahiert und als Import verwendet werden können. |
| Prüfungszeitfenster festlegen | Es gibt eine Möglichkeit, ein Prüfungsfenster zu definieren. Ein Starten der Prüfung ausserhalb dieses Zeitfensters darf nicht möglich sein. |
| Prüfungsdauer festlegen | Durch einen Konfigurationsparameter kann eingestellt werden, wie lange die Prüfung dauert. Nachdem die Zeit abgelaufen ist, wird die Prüfung automatisch beendet. |
| Log schreiben | Im Hintergrund wird ein log geschrieben, welches festhält, wie oft die Prüfung gestartet wurde und Manipulationsversuche dokumentiert. |
| AutoSave Funktion | Um bei einem Absturz des Gerätes keine Daten zu verlieren, soll der aktuelle Zwischenstand der Prüfung periodisch gespeichert werden. |
| Sicherheitsfeatures konfigurierbar | Die verschiedenen Sicherheitsfeatures sollen mittels einer Konfigurationsdatei individuell ein- und ausgeschalten werden können. |
| Tabwechsel detektieren | Es soll festgestellt werden können, wenn ein Prüfungskandidat auf einen anderen Tab wechselt. |
| Page-Mode für eBook Reader | Scrollen ist auf eBook Readern sehr langsam, deshalb soll ein Page-Mode entwickelt werden. In diesem Modus wird immer nur eine Frage angezeigt und man kann mittels Weiter/Zurück die Frage wechseln. |
| Parameterisierbare Verschlüsselungsstärke | Die Stärke der Verschlüsselung bzw. die Länge des Passworts kann mittels eines Parameters oder einer Konfigurationsdatei definiert werden. |

## Zielsetzung

Ziel der Arbeit ist es, ein Programm zu entwickeln welches eine verschlüsselte Prüfung generiert. Die Prüfung wiederum besteht aus einer Datei, welche auf einen eBook Reader geöffnet werden kann. Bei dem Start der Prüfung muss sich der Student mit einem Key authentifizieren, welcher zum Entschlüsseln verwendet wird. Die Prüfung wird nach dem Login automatisch und transparent für den Benutzer entschlüsselt und gestartet. Es muss sichergestellt werden, dass die Prüfung nach dem Abgeben nicht mehr manipuliert werden kann und der Dozent muss die Prüfungsdaten auf einfache Art und Weise von dem eBook Reader extrahieren können.

## Zielpublikum

Die Arbeit umfasst zwei Teile, den Prüfungsgenerator und die generierte Prüfung, welche jeweils ein eigenes Zielpublikum besitzen:

**Prüfungsgenerator:**

Die Applikation soll von Dozenten der ZHAW verwendet werden, womit diese auch als Zielpublikum definiert sind. Sie besitzen ein fundiertes, technisches Know-How und können mit standardisierten Dateiformaten umgehen.

**Prüfung:**

Zielpublikum für die Prüfung sind Studenten der ZHAW oder Absolventen der Aufnahmetests. Beide können Geräte mit Touchscreen und einer Touch-Tastatur einwandfrei bedienen.

# Grundlagen

## Kryptographie

### Verschlüsselung

Die Verschlüsselung eines Datensatzes dient generell dazu, dessen Inhalt vor unberechtigten Personen zu verbergen (Confidentiality). Die zwei am weitest verbreitetsten Methoden sind die Secret Key Verschlüsselung und die Public Key Verschlüsselung. Anschliessend wird jeweils der aktuelle Standard beschrieben.

#### AES

AES wurde im Jahre 2001 offiziell publiziert und ist der momentane Standard bei der „Secret Key“ Verschlüsselung. AES verschlüsselt den gegebenen Input nicht als Ganzes, sondern unterteilt diesen in kleine Blöcke (128 bis 258 Byte), welche letztendlich Stück für Stück verschlüsselt werden.



Abbildung 1: AES Block Verschlüsselung

Für die Verschlüsselung der Blöcke stehen zwei verschiedene Modi zur Verfügung:

|  |  |
| --- | --- |
| ECB    Abbildung 2: ECB Verschlüsselungsmodus  Jeder Block wird unabhängig betrachtet und mit dem gleichen Schlüssel verschlüsselt. Ausserdem unterstützt der ECB Mode per Definition keinen IV. | CBC    Abbildung 3: CBC Verschlüsselungsmodus  CBC Verschlüsselt dem ersten Block mit einem Initialisierungsvektor (IV) und die folgenden Blöcke mit dem Ciphertext des jeweils vorherigen Blockes. |

CBC bietet im Vergleich zu ECB eine erhöhte Sicherheit gegen Manipulationen am Ciphertext. Dadurch, dass die Blöcke abhängig voneinander Verschlüsselt werden, würde beim Austauschen eines Blockciphers der jeweils vorherige beschädigt. (vgl. Rennhard, 2014)

#### RSA

RSA ist der Standard bei der Public / Private Key Verschlüsselung. Dies bedeutet, dass für jeden Teilnehmer ein Private - sowie Public Key generiert werden muss. Mittels Signierung der Zertifikate von einer global anerkannten „Certificate Authority“ kann zusätzlich die Echtheit eines Zertifikates ermittelt werden.



Abbildung 4: RSA Verschlüsselung

User 2 Verschlüsselt seine Nachricht an User 1 mittels dem frei Verfügbaren Public Key von User 1 und überträgt anschliessend die verschlüsselte Nachricht an den Empfänger. User 1 kann nun die Nachricht mit seinem Private Key entschlüsseln und die Originalnachricht lesen. (vgl. Rennhard, 2014)

### Passwort Hashing

Kryptographische Hashfunktionen berechnen aus einem variabel langen Input einen fixen Output (den Hash). Dieser muss folgende Eigenschaften aufweisen:

* Pseudo zufällig: Wenn ein Bit des Inputs verändert wird, müssen sich ca. 50% der Outputbits auch ändern.
* One Way: mit einem gegebenen Hash muss es praktisch unmöglich sein, die Ursprungs-nachricht zu ermitteln.
* Collision free: Es muss praktisch unmöglich sein, zwei Nachrichten zu finden, welche den selben Hash produzieren.

Typische Hash-Längen (Output Länge) sind je nach gewähltem Algorithmus zwischen 128 und 512 Bits lang.

Beim Passwort Hashing wird oft ein Hash-Chaining angewendet. Dies bedeutet, dass die Hash-funktion mehrmals nacheinander ausgeführt wird, um die Berechnungszeit des Hashs zu erhöhen. Ein Brute-Force Angriff auf den Hash dauert, je mehr Hash-Iterationen gemacht werden, entsprechend länger. (vgl. Rennhard, 2014)

#### SHA3

Die momentan neuste Version der SHA Algorithmen, welche vom US-Amerikanischen National Institute of Standards and Technology (NIST), veröffentlicht werden. Die verfügbaren Hashlängen sind zwischen 224 und 512 Bit, was eine Sicherheit gegen Kollisionsangriffe von 256 Bit bietet (wegen des Geburstagsparadoxum). (vgl. NIST, 2014)

#### PBKDF2

PBKDF2 steht für Password Based Key Derivation Function und wurde entwickelt, um von einem Passwort einen Key abzuleiten, welcher in einem symmetrischen Verschlüsselungsverfahren verwendet werden kann.

PBKDF2 wendet auf den Input einen Hashalgorithmus mit Salting und anschliessendem Chaining an. (vgl. Kaliski, 2000)

## Dateiformate

### INI

Eine Art der Konfiguration ist mittels INI-Datei möglich. Eine solche Initialisierungsdatei ist eine Textdatei, welche es ermöglicht verschiedene Wertepaare abzulegen und zu definieren. Die Wertepaare bestehen aus einem Schlüssel und einem Wert. Schlüssel und Wert werden durch ein Gleichheitszeichen zusammengesetzt.

### XML

Die Konfiguration von Initialisierungsparametern und der Import von Prüfungsdetails kann mithilfe einer XML-Datei umgesetzt werden. Es dient zur hierarchischen Strukturierung von Daten und ist plattformunabhängig. Damit das Dokument als gültig betrachtet werden kann, muss es mehrere Regeln (vgl. Bray, et al., 2006) erfüllen, als da wären:

* Es besteht genau ein äusserstes Element, das so genannte Wurzelelement.
* Zu jedem Start-Tag gehört ein End-Tag, dass identisch geschrieben werden muss, es sei denn, das Element hat keinen Inhalt, dann kann auch das Element in sich geschlossen werden.
* Alle Elemente müssen richtig verschachtelt sein.
* Jedes Attribut eines Elements muss einen eindeutigen Namen besitzen.
* Attributeigenschaften werden in Anführungszeichen angegeben.

Mithilfe eines Parsers werden valide XML-Dateien eingelesen, geprüft und einer neuen Struktur übergeben, mit welcher die Daten weiter verwendet werden können.

### Open Office

Mithilfe von Open Office ist es möglich eine Prüfung zu erstellen und diese als ODT-Datei abzulegen. Das ODT-Format ist nichts anderes, als ein komprimierter Ordner, der die Inhalte der Datei sauber gegliedert im XML-Format enthält. Der Inhalt dieser XML-Datei basiert auf den Elementen von HTML.



Abbildung 5: Inhalt einer ODT-Datei

### Microsoft Word

Microsoft Word ermöglicht es, durch einfache Eingabe eine Prüfung zusammen zu stellen und diese in verschiedenen Formaten zu speichern. Die Struktur von Word-Dokumenten ist ebenfalls im XML-Format, welches auf HTML basiert. Ein Word Dokument ist ein komprimierter Ordner, dessen Aufbau nach dem Entpacken sichtbar wird. Es beinhaltet ein XML-File mit den Inhaltsangaben und Verweisen, die eingefügten Steuerelemente werden in Ordnern innerhalb der Datei abgespeichert.



Abbildung 6: Inhalt einer DOCX-Datei

### XSLT

XSLT ist eine Programmiersprache zur Umwandlung von XML-Dateien in andere Formen von XML-Dokumenten oder XML-ähnliche-Formate. Das Format von XSLT ist selbst in der XML-Struktur definiert und kann einfach mit einem Texteditor erstellt und bearbeitet werden.

Zur Erstellung einer XML-Struktur mithilfe von XSLT wird eine zusätzliche Programmiersprache benötigt, wie zum Beispiel C#.Net. In dieser Applikation wird dann sowohl das XML File, welches zur Eingabe gedacht ist, als auch die definierte XSLT-Datei importiert. Nachdem die XSLT-Funktion die auszugebende XML-Struktur erstellt hat, wird diese Ausgabe an einen vordefinierten Ort gespeichert und kann für weitere Arbeiten verwendet werden.

Abbildung : Funktionsweise von XSLT

### PDF

Eine mögliche Form zur Exportierung bietet das PDF-Format an. Es handelt sich dabei um eine Textstruktur, welche sich durch die Verwendung von Bibliotheken generieren lässt und die Verwendung von Steuerelementen erlaubt. PDF hat die Eigenschaft, dass es Plattformunabhängig ist, was für manche Projekte von Vorteil sein kann.

### HTML mit Javascript

Eine weitere Art die es ermöglicht exportierte Daten in einem verwendbaren Format darzustellen, ist HTML. Durch die Verwendung von vorbereiteten Skeletons lässt sich eine HTML-Datei erzeugen, welche dann als Prüfung verwendet werden kann. Die Interaktionen zwischen dem Benutzer und der Seite werden mithilfe von Javascript umgesetzt. HTML und Javascript sind Plattformunabhängig und es lassen sich damit unterschiedliche Projekte realisieren.

## Digitale Manipulationsmöglichkeiten

Sobald Prüfungen in digitaler Form abgelegt werden können, gibt es immer Bedenken betreffend Manipulationsmöglichkeiten während bzw. an der Prüfung. Nachfolgend sind die möglichen Manipulationen aufgelistet, die bei Prüfungen auf eBook Readern auftreten könnten.

### Korrekte Antwort auslesen

Für die Korrektur der Antworten sind bereits die korrekten Antworten irgendwo in der Prüfung gespeichert, wenn auch nicht sichtbar für den Prüfungsteilnehmer. Es wäre also mit genügend Fachwissen möglich, diese auszulesen und somit die Prüfung vollständig korrekt zu beantworten.

### Zeit

Die Restzeit der Prüfung wird digital berechnet. Falls es nun möglich wäre die Zeit zu manipulieren, könnte sich ein Teilnehmer eine grössere Zeitspanne für die Prüfung ermöglichen. Um dies zu realisieren wären folgende zwei Szenarien möglich:

#### Manipulation der geräteinternen Uhr

Angenommen die Zeitspanne bis zum Ende der Prüfung berechnet sich mittels folgender Formel:

Dann wäre es möglich, nach dem Starten der Prüfung, die interne Uhr um eine gewisse Zeitspanne zurückzusetzen um dem Kandidat mehr Zeit für die Prüfung zu generieren.

#### Verlangsamung der Zeit

Um die Zeitmanipulation besser zu tarnen, könnte in regelmässigen Abständen, die Zeit um ein paar Millisekunden zurückgesetzt werden. Angenommen, man würde alle 10ms die Zeitrechnung um 5ms zurücksetzen, dann würde eine Sekunde der Systemuhr plötzlich zwei reale Sekunden dauern. Der Kandidat hätte also die doppelte Zeitspanne zum Lösen der Prüfung zur Verfügung.

### Internet Zugriff

Dadurch dass fast alle Geräte heutzutage mit Wireless Modulen ausgestattet sind, wäre es möglich, während der Prüfung im Internet nach Lösungen der Prüfungsfragen zu suchen.

### Prüfungsinterne Kommunikation

Sobald technische Geräte mit Kommunikationsfunktionen ausgestattet sind, können diese auch missbraucht werden. Probanden hätten somit die Möglichkeit, während der Prüfung untereinander zu kommunizieren. Aktuelle eBook Reader könnten folgende Kommunikationsmodule verbaut haben:

#### Wireless LAN

Auch wenn der Prüfungsraum gegen jegliche Wirelessstrahlung von aussen abgeschirmt wäre, wäre es trotzdem möglich, dass intern Ad-Hoc Netzwerke erstellt würden. Ein Teilnehmer könnte ein solches erstellen und die anderen Probanden könnten diesem beitreten und darüber kommunizieren.

#### Bluetooth Modul

Mittels Bluetooth sind Datenverbindungen zwischen zwei Endgeräten möglich. Somit könnten Kandidaten durch einschalten von Bluetooth miteinander Daten austauschen / miteinander kommunizieren.

## eBook Reader

Als eBook Reader werden technische Geräte bezeichnet, welche primär zum Lesen von Büchern erstellt und optimiert wurden. Für die Eigenschaften der Geräte bedeutet dies, dass der Fokus auf langer Akkulaufzeit und angenehmem lesen von Texten auf dem Bildschirm liegt.

Um eine lange Akkulaufzeit zu erreichen, werden oft eher langsame elektronische Komponenten verbaut und es werden eingeschränkte, speziell angepasste Betriebssysteme verwendet. Es wird nachfolgend generell davon ausgegangen, dass eBook Reader kein voll funktionsfähiges Android OS besitzen. Solchen Geräte unterscheiden sich kaum mehr von Tablets und würden als solche angesehen.

### Einschränkungen

#### Betriebssystem

Die Betriebssysteme von eBook Readern sind meist vereinfachte Versionen von Android (z.B. Tolino OS) oder vom Hersteller selber entwickelte Linux Distributionen(z.B. nicht Android basierte Amazon Kindle Geräte). Die Idee dahinter ist, ein einfaches, Ressourcen schonendes System zu haben, welches gerade nur die Funktionen bietet, die zum Lesen von eBooks benötigt werden.

#### CPU / GPU Geschwindigkeit

EBook Reader benötigen für den normalbetrieb sehr wenig Rechengeschwindigkeit. Die CPU und GPU werden so gewählt, dass in akzeptabler Zeit eine Buchseite dargestellt werden kann und dass der Endbenutzer eine möglichst lange Akkulaufzeit hat. Es steht somit deutlich weniger Rechenleistung zur Verfügung als bei aktuellen Tablets.

#### Displaytechnik

Bei eBook Readern werden im Gegensatz zu den Tablets sogenannte E-Ink Displays verwendet. Diese Displays brauchen nur Strom, wenn das Bild geändert wird. Eine Änderung wäre zum Beispiel das wechseln von einer Buchseite auf eine andere. Ein statisches Bild bleibt ohne zusätzlich benötigte Energie über einen, je nach Display variierenden, Zeitraum erhalten. Der Fokus der Technik liegt ganz klar bei der Energieeffizienz und der guten Lesbarkeit bei möglichst vielen Lichtverhältnissen.

# Analyse

## Anwendungsfälle

### Übersicht

* AF01: Prüfung erstellen
* AF02 Prüfung absolvieren

### Akteure

#### Dozent

Der Dozent ist ein Akteur, welcher die Berechtigung hat, eine Prüfung zu erstellen und die einzelnen Angaben zu den Studenten kennt. Dies kann auch ein Assistent des Dozenten sein.

#### Student

Alle Personen, die eine Prüfung durchführen müssen, werden in den Anwendungsfällen als Student bezeichnet.

#### System

Als System wird die Applikation selbst bezeichnet.

### AF01: Prüfung erstellen

#### Kurzbeschreibung

Dem Benutzer muss es möglich sein, die Prüfung zu erstellen. Falls die Dateien, welche vorher angegeben werden müssen, korrekt sind, werden eine einzelne Prüfungsdatei und das Passwort-File an dem Ort erstellt, welcher definiert wurde.

#### Akteure

Dozent, System

#### Auslöser

Sobald der Dozent die Konsolenapplikation aufruft, wird die Prüfungserstellung gestartet.

#### Vorbedingung

Die Person, welche die Prüfung erstellen möchte, hat alle Dateien zur Übergabe gemäss Bedienungsanleitung vorbereitet. Alle Konsolenparameter sind richtig gesetzt und enthalten keine Fehler.

#### Ergebnisse und Nachbedingungen

Die Prüfung wurde mithilfe von den Konfigurationen erstellt und steht als exportierte HTML-Datei zur Verfügung. Die generierten Passwörter bestehen aus zufälligen Zahlen, Buchstaben und Sonderzeichen. Die Login-Informationen sind in einer separaten Datei abgespeichert.

#### Benutzte Anwendungsfälle

Keine

#### Ablauf

|  |  |  |
| --- | --- | --- |
| Schritt | Akteur | Ablauf |
| 1. | Dozent | Eingeben der Parameter |
| 2. | Dozent | Prüfung erstellen |
| 3. | System | Importiert und verarbeitet Dateien |
| 4. | System | Generiert Prüfung und Passwörter |
| 5. | System | Exportiert die verschlüsselte Prüfung und die Login-Daten |
| 6. | System | Beendet Programm |
|  |  | **Alternativer Ablauf** |
| 2.a | System | Error Meldung wird ausgegeben: Fehler in Parameter |
| 2.b | System | Erstellen wird abgebrochen |
| 3.a | System | Error Meldung wird ausgegeben: Fehler in Importdatei |
| 3.b | System | Erstellen wird abgebrochen |

#### Spezielle funktionale Anforderungen

Die Parameter müssen klar definiert sein und wenn ein Fehler auftritt, muss eine aussagekräftige Meldung erscheinen. Wenn die Konsolenparameter einen Fehler enthalten muss die richtige Verwendung dieser Applikation angezeigt werden.

#### Spezielle technische Anforderungen

Das Erstellen der Prüfung darf nicht länger als zwei Sekunden dauern, wenn es länger dauern würde, wäre die Applikation nicht benutzerfreundlich.

#### Häufigkeit

Jedes Mal wenn eine Prüfung mit dieser Applikation erstellt werden soll, wird dieser Anwendungsfall mindestens einmal gebraucht, falls Anpassungen gemacht werden müssen oder verschiedene Versionen der Prüfung entstehen sollen, wird dieser Anwendungsfall mehrmals durchgelaufen.

#### Datenspeicherung

In diesem Anwendungsfall werden zwei Dateien gespeichert, welche relativ klein sind. Zum einen die generierte Prüfungsdatei und zum anderen die Login-Daten für die Studenten.

#### Priorität

**Hohe Priorität**

Es handelt sich um den Hauptanwendungsfall, der mit höchster Priorität behandelt werden muss.

### AF02 Prüfung absolvieren

#### Kurzbeschreibung

Mit der generierten Prüfung, können Studenten ihre Fähigkeiten beweisen. Sie öffnen die Prüfungsdatei und können nach dem Login mit der Prüfung beginnen. Am Ende der Prüfung werden die Resultate abgegeben.

#### Akteure

Student, System

#### Auslöser

Das Öffnen der Prüfungsdatei zeigt dem Studenten das Anmeldeformular an. Nach dem Login beginnt die Prüfung.

#### Vorbedingung

Die Prüfungsdatei wurde von dem zuständigen Dozenten erstellt(Siehe Anwendungsfall „AF01: Prüfung erstellen“) und mit einem fähigen Gerät ausgeliefert. Der Student hat sein Passwort für die Prüfung vom Dozenten erhalten.

#### Ergebnisse und Nachbedingungen

Die Prüfung wurde vom Studenten ausgefüllt oder hat die Antworten leer gelassen. Die Daten wurden verschlüsselt und exportiert.

#### Benutzte Anwendungsfälle

„AF01: Prüfung erstellen“

#### Ablauf

|  |  |  |
| --- | --- | --- |
| Schritt | Akteur | Ablauf |
| 1. | Student | Öffnen der Prüfungsdatei |
| 2. | Student | Einloggen |
| 3. | Student | Lösen der Prüfung |
| 4. | Student | Abgabe der Prüfung |
| 5. | System | Prüfungsantworten werden exportiert |
|  |  | **Alternativer Ablauf** |
| 2.a | Student | Einloggen gescheitert |
| 2.b | System | Error Meldung wird ausgegeben |
| 2.c | System | Anmeldemaske wird wieder angezeigt |
| 3.a | System | Manipulation erkannt |
| 3.b | System | Prüfung wird beendet und die Antworten exportiert |
| 4.a | System | Zeit abgelaufen |
| 4.b | System | Prüfung wird beendet und die Antworten exportiert |

#### Spezielle funktionale Anforderungen

Das Programm sollte intuitiv durch den Studenten bedienbar sein. Alle Beschriftungen und Meldungen müssen klar und deutlich formuliert sein.

#### Spezielle technische Anforderungen

Der Anmeldevorgang der Prüfung darf nicht länger als drei Sekunden dauern, wenn es länger dauern würde, wäre das Programm nicht benutzerfreundlich.

#### Häufigkeit

Jeder Student, der eine Prüfung absolvieren muss, durchläuft diesen Anwendungsfall pro Prüfung genau einmal.

#### Datenspeicherung

In diesem Anwendungsfall werden die Prüfungsantworten exportiert und gespeichert. Die Datenmenge ist sehr klein.

#### Priorität

**Hohe Priorität**

Es handelt sich hierbei um einen Anwendungsfall, der direkt mit dem Hauptanwendungsfall in Verbindung steht und für das Projekt unumgänglich ist.

## Nichtfunktionale Anforderungen

### NFA01 Prüfung ist einfach bedienbar

Die Prüfung soll intuitiv von den Studenten bedient werden können. Um dies zu erfüllen, muss das Prüfungsprogramm einfach gehalten werden, es darf nicht überladen sein und alle Beschriftungen und Meldungen müssen einfach, verständlich und eindeutig sein.

### NFA02 Export erfolgt transparent für den Probanden

Der Student soll nach der Prüfung keine Einsicht mehr in die Prüfung haben und diese auch nicht mehr verändern können. Die Daten müssen deshalb verschlüsselt abgespeichert werden, was der Student nicht mitbekommen soll, da ihm, ab diesem Zeitpunkt, die Prüfung nicht mehr gehört.

### NFA03 Anmeldevorgang dauert weniger als drei Sekunden

Da die Prüfungsdatei zu Beginn noch verschlüsselt ist, muss sie beim Anmeldevorgang für den Studenten entschlüsselt werden. Eine Entschlüsselung kann sehr Aufwändig sein und verursacht deshalb oft eine Wartezeit. Aufgrund dessen, dass eine lange Wartezeit nicht als Benutzerfreundlich gilt muss die Verschlüsselung so gewählt werden, dass diese Zeitspanne nicht mehr als drei Sekunden betragen darf.

# Konzept

## Systemüberblick



Abbildung 8: Systemüberblick

Das Diagramm bietet eine Übersicht über die Komponenten. Die Applikation SecureExam.exe generiert aus den drei Eingabedateien die Prüfung sowie eine Datei mit den Passwörtern für die Studenten. Beim Öffnen der Prüfungsdatei wird der Student aufgefordert, seine erhaltenen Daten (Vorname, Nachname, Immatrikulationsnummer, Zufallspasswort) einzugeben. Sobald diese korrekt eingegeben wurden, entschlüsselt sich die Prüfung selber und die Fragen werden angezeigt. Beim Abschluss der Prüfung werden die Antworten exportiert.

### Domänenmodell



Abbildung 9:Domänenmodell SecureExam

## C# Errorhandling

Damit ein einheitliches Errorhandling entsteht, werden im ganzen Code keine Exceptions abgefangen, ausser es wird zur besseren Übersicht eine andere Exception ausgelöst (Exception rethrowing).



Abbildung 10: Beispiel eines Exception rethrowings

Die Exceptions werden global in der Hauptapplikation abgefangen, da diese Zugriff auf die GUI Elemente hat und den Benutzer im Fehlerfall direkt informieren kann. Es entsteht so zudem eine saubere Trennung zwischen GUI und Logik und ergänzt die Modularität von SecureExam im Bezug auf neue GUI-Implementationen.

## Codedokumentation

Jede Funktion im Code wird mittels standardisierten Kommentaren dokumentiert. So können die Kommentare zusätzlich direkt in der IDE für die Code-Vervollständigung verwendet werden.

### C#

Der C#-Code wird mittels Kommentaren und den XML Kommentartags, welche von C# zur Verfügung gestellt werden dokumentiert. Die Dokumentation wird exportiert und steht danach als eigenständige Datei zur Verfügung.

### JavaScript

JavaScript hat per default kein Dokumentationsframework eingebaut wie C#. Deshalb wird der Code mittels der Dokumentations API „JSDoc“ dokumentiert. Die Kommentare können mit dem JSDoc Tool exportiert werden, womit dann eine eigenständige Dokumenation in HTML Format zur Verfügung steht.

## Kryptographie

### Verschlüsselung

#### Evaluationsmatrix



Tabelle 1: Vergleich AES und RSA

#### Entscheid

Wie bei der Evaluationsmatrix(vgl. **Fehler! Verweisquelle konnte nicht gefunden werden.**) ersichtlich, hat AES gegenüber RSA einen grossen Vorteil in Bezug auf die „einfache Keyeingabe“. Bei AES kann man ein normales Passwort mit anschliessendem Hashing verwenden, bei RSA müsste ein komplettes Zertifikat korrekt abgetippt werden. Deshalb wird AES als Verschlüsselungsverfahren eingesetzt.

#### Konzept

Für eine bessere Übersicht wird der Ver- bzw. Entschlüsselungsvorgang getrennt beschrieben. Die Verschlüsselung wird automatisch mittels dem Prüfungsgenerator durchgeführt, die Entschlüsselung hingegen findet auf dem Client direkt beim Starten der Prüfung statt.

##### Verschlüsselung



Abbildung 11: Multi-User AES Verschlüsselung

Für jeden Student wird ein individuelles Passwort generiert. Dieses setzt sich aus folgenden Komponenten zusammen:

* Vorname
* Nachname
* Immatrikulationsnummer
* n zufällige Zeichen (Base64)
* 128 Bit Salt.

Diese werden anschliessend n-Mal mit gehasht (vgl. 4.4.2, Passwort hashing). Mit dem resultierenden Hash (256 Bit) und einem zufälligen IV (Initialisierungsvektor) wird nun der Masterkey mittels AES verschlüsselt. Der Masterkey setzt sich aus einer zufälligen, 256 Bit langen Zeichenkette zusammen. Dieser wird dazu verwendet, um die Prüfungsdaten (Fragen, Prüfungszeitraum und Prüfungsdauer) zu verschlüsseln.

##### Entschlüsselung



Abbildung 12: Multi-User AES Entschlüsselung

Jeder Student wird beim Starten der Prüfung seine Zugangsdaten eingeben müssen. Diese bestehen aus:

* Vorname
* Nachname
* Immatrikulationsnummer
* n zufällige Zeichen

Mit diesen Informationen plus dem zugehörigen Salt, wird wie bei der Verschlüsselung, ein Hashing durchgeführt. Das resultierende „Student Password“ dient als Passwort für die Entschlüsselung des Masterkeys. Mittels diesem und dem zugehörigen IV, werden dann die eigentlichen Daten entschlüsselt.

### Passwort hashing

#### Evaluationsmatrix



Tabelle 2: Vergleich SHA3 und PBKDF2

#### Entscheid

Die beiden Algorithmen sind praktisch gleichauf. PBKDF2 ist ein Algorithmus, welcher absichtlich langsam, für das generieren eines Keys aus einem Passwort entwickelt wurde. Wie in der Evaluationsmatrix ersichtlich, wird SHA3 als Hash-Algorithmus eingesetzt. Der Entscheid wurde nur aufgrund der Skalierbarkeit getroffen, SHA3 ist schneller (benötigt weniger Rechenzeit), weshalb eine bessere Anpassung an langsame Hardware (eBook Reader) gemacht werden kann.

#### Hashing Konzept

Um die Zeit, welche zum Hashen benötigt wird, optimal an die eBook Reader Hardware anzupassen, ist die Anzahl Iterationen beim Chaining Vorgang via Konfigurationsparameter anpassbar.



Abbildung 13: Hashing Ablauf

Die Studenteninformationen werden zusammen mit n zufälligen Zeichen (Passwort) und einem kryptografisch sicheren, zufälligen Salt gehasht. Der resultierende Hash wird noch (n-1) Mal mittels SHA3 gehasht und der letzte Hashwert wird als Resultat ausgegeben und als Key verwendet.

## Konfigurationsdateien

### Entscheid

Aufgrund der Tatsache, dass zum Zeitpunkt der Evaluation des Konfigurationsdateityps bereits Fortschritte mit dem Parsen von XML gemacht wurden, wäre es nur zu mehr Aufwand gekommen, wenn eine INI-Datei hätte eingesetzt werden müssen. Des Weiteren ist es mit XML möglich eine nützliche und lesbare Struktur zu definieren und bietet keine Nachteile zur Verwendung einer INI-Datei. Deshalb wurde der Entscheid gefällt, XML einzusetzen.

### Konzept

Es werden zwei verschiedene Konfigurationsdateien verwendet, um Einstellungen bezüglich der Prüfung von den Einstellungen des Programmes abzugrenzen. Beide Dateien sind im XML-Format definiert, da dies so entschieden wurde.

## Import

### Evaluationsmatrix



Tabelle 3: Vergleich ODT und Word

### Entscheid

Sowohl Microsoft Word sowie auch Open Office verwenden die XML-Struktur, welche sich grundsätzlich für die Verwendung in diesem Projekt eignen würde, jedoch sind Elemente wie Checkboxen und Textfelder in Word so verschachtelt, dass das Parsen eines solchen Files schwieriger zu realisieren wäre, als das Parsen eines ODT Dokuments. Der Aufbau eines Open Office Dokuments eignet sich besser für unser Projekt, daher wurde entschieden einen Parser für ODT einzubauen.

### Konzept

Es werden zwei verschiedene Methoden Angeboten die Prüfungsfragen zu importieren. Die eine Lösung wird sein, ein XML-File zu importieren, da die Struktur bereits vorhanden ist, muss eine solche Datei nur noch mit einem Parser eingelesen werden.

Die zweite Variante wird sein, ein ODT-Dokument anzugeben, welches mithilfe von XSLT zu einer verwertbaren XML-Struktur verarbeitet wird und dann mithilfe des bereits erwähnten XML-Parsers eingelesen werden kann.

Zusätzlich müssen die Angaben zu den Studenten, welche die Prüfung absolvieren sollen noch importiert werden. Da sich die XML-Struktur am besten dazu eignet, wird ein zusätzlicher XML-Parser gebraucht, welcher diese Arbeit ausführt.

## Export

### Evaluationsmatrix



Tabelle 4: Vergleich HTML und PDF

### Entscheid

Die Plattformunabhängigkeit wird für dieses Projekt als wichtig eingestuft, da es den Benutzern der Software die Freiheit gibt, einzusetzende Geräte und Betriebssysteme selber zu wählen. Beide Exportvarianten, welche evaluiert wurden, haben dieses Kriterium erfüllt. Nach Absprache mit dem betreuenden Dozenten wurde aufgrund der Erfahrungen im Team, welche bei HTML mehr ausgeprägt sind, wie bei der Softwareentwicklung mit PDF, die Exportvariante HTML gewählt.

### Konzept

Das Exportieren von Daten in eine HTML-Struktur funktioniert am besten über so genannte Skeletons. Diese Grundgerüste beinhalten bereits viele vordefinierte Elemente sowie CSS und JavaScript Funktionen. Die Idee besteht darin, die ganze Prüfung in nur einer Datei zu speichern.

Die Login Daten müssen natürlich separat gespeichert werden, da diese zum Entschlüsseln der Prüfung gebraucht werden. Die Idee ist deshalb, diese in ein anderes File mit einem brauchbaren Format zu exportieren.

## Digitale Manipulationsmöglichkeiten

### Zeit

#### Manipulation der geräteinternen Uhr

Um eine Änderung bei der geräteinternen Uhr feststellen zu können, wird beim Starten der Prüfung automatisch ein Zeitverlauf angelegt. D.h. jede Sekunde wird die aktuelle Systemzeit ausgelesen und in ein Array abgespeichert. Anschliessend wird überprüft, ob die neuste Uhrzeit eine zu grosse Varianz gegenüber den alten besitzt (Varianz ist einstellbar via Settings Datei).



Abbildung 14: Manipulation der internen Uhr detektieren

Wenn nun also ein Student die Geräteinterne Uhr während der Prüfung um z.B. 20 Minuten zurücksetzt, wird die Prüfung eine Varianz von -20 Minuten gegenüber der vorherigen Uhrzeit feststellen und die Prüfung abbrechen.

#### Verlangsamung der Zeit

Theoretisch könnte diese Manipulation genauso detektiert werden wie oben. Im Prinzip unterscheiden sich die beiden Manipulationen nur dadurch, dass bei der Verlangsamung die Uhr periodisch um wenige Millisekunden zurückgesetzt wird gegenüber einer grossen Veränderung. Es wäre möglich, die Uhr einfach öfters auszulesen um auch kleinere Abweichungen erfassen zu können, dies würde aber zur Folge haben, dass die Methode „Zeit auslesen“ öfters aufgerufen werden müsste. Auf eBook Readern gibt es nur beschränkte Rechenleistung, deshalb wird ein neues Konzept angewandt:



Abbildung 15: Vergleich der internen Uhr mit der Systemuhr

Nachdem die Prüfung gestartet wird, wird eine Prüfungsinterne, in JavaScript realisierte Uhr gestartet. Diese läuft parallel zu der Systemuhr und somit kann man Abweichungen zwischen den zwei Zeiten sehr einfach feststellen.

### Internet Zugriff

Durch periodischen Versuch, ein Bild aus dem Internet herunterzuladen, wird überprüft ob das Gerät über eine aktive Internetverbindung verfügt. Solange es beim Einbindungsversuch einen Fehler gibt, ist die Verbindung offline, wenn nicht, wird vom System ein Event ausgelöst und die Prüfung je nach Konfiguration abgebrochen.

### Prüfungsinterne Kommunikation

Generell ist es technisch nicht möglich, mittels JavaScript festzustellen, ob sich ein Gerät in einem Ad-Hoc Netzwerk befindet oder gerade Daten über Bluetooth austauscht. Dazu müsste eine Prüfungs-App entwickelt werden, welche die nötigen Berechtigungen besitzt, um die entsprechenden Adapter zu überwachen.

Trotzdem kann der internen Kommunikation durch geschickte Wahl der eBook Reader entgegen gewirkt werden. Vorzugsweise werden eBook Reader eingesetzt, welche keine WLAN Ad-Hoc Funktionalität besitzen und kein Bluetooth Modul verbaut haben.

## UI-Design

### Idee

SecureExam virtualisiert die Prüfung, die Papierform wird überflüssig und Prüfungen können digital abgelegt werden. Inspiriert von der alten Form der Prüfung, auf Papier, wird SecureExam mittels Material Design umgesetzt. Den Prüfungsabsolventen wird eine gewohnte Umgebung vorgespielt, damit sie sich während der Prüfung wohler / entspannter fühlen. Durch neue Möglichkeiten, welche durch die Digitalisierung mit sich kommen, werden die Prüfungsblätter um nützliche Funktionen, wie zum Beispiel die Anzeige der verbliebenen Prüfungszeit, ergänzt.



Abbildung 16: UI Design Idee

Auf der Handskizze oben, sind zwei Blätter zu erkennen. Das graue Blatt im Querformat wird oben an der Prüfung sein und die wichtigen Informationen (Titel, Prüfungshinweise, Restzeit etc.) beinhalten. Es ist horizontal gefaltet um eine visuelle Trennung zwischen Titel und Prüfungshinweisen zu gestalten. Das schwarze Blatt im Portraitformat ist das Prüfungsblatt und beinhaltet die Prüfungsfragen.

### Login



Abbildung 17: Login-Skizze

Das Erste wo man von der Prüfung sieht, ist der Login-Screen. Hier muss sich der Student authentifizieren. Es gibt total vier Eingabefelder, die der Prüfungsteilnehmer ausfüllen muss:

* Vorname
* Name
* Immatrikulationsnummer
* Passwort

Nach dem Klick auf Starten, startet der Entschlüsslungsvorgang und die Ansicht wechselt je nach Einstellung in den Page- oder Scrollmodus, wo die entschlüsselten Fragen dargestellt werden.

### Prüfung

#### Page-Mode



Abbildung 18: Skizze Paging Mode

Der Page Mode ist dafür gedacht, wenn die Prüfung auf eBook Readern mit sehr schlechter Bildwiederholfrequenz ausgeführt wird. Zwischen den Fragen wird, ähnlich wie bei einem Buch, geblättert. Somit wird die Problematik des Scrollens (Bild wird verspätet dargestellt) umgangen.

#### Scroll-Mode

Der Scroll-Mode wird eingesetzt, wenn die Prüfung auf eBook Readern mit LCD Display oder sogar Tablets abgelegt wird. Im Gegensatz zum Page-Mode, sind jederzeit alle Fragen sichtbar. Die Navigation zwischen den Fragen ist wie von vielen anderen Anwendungen gewohnt, mittels Scrollen möglich.

Damit der Student jederzeit die Übersicht über die verbliebene Zeit und diverse andere wichtige Information behält. Scrollt der gefaltete Teil des grauen Blattes mit Titel etc. konstant mit.

Abbildung 19: Skizze Scroll Mode

## JavaScript

### SecureExam Library

[Klassendiagramm]

### CryptoJS Library

CryptoJS ist eine sehr modular aufgebaute JavaScript Library für kryptographische Funktionen und ist unter der „New BSD License“ frei verfügbar. In SecureExam wird CryptoJS für folgende Funktionalitäten verwendet:

* SHA3
* AES
* Bit / Byte Datenstrukturen

Gerade durch die Modularität eignet sich die Library perfekt für den Einsatz in SecureExam, da nur genau die Module in das Prüfungsfile eingebunden werden müssen, die auch wirklich benötigt werden.

### FileSaver.js Library

Zum Exportieren in der „nicht eBook Version“, wird aus den ausgefüllten Feldern automatisch ein verschlüsseltes Dokument erstellt, welches dann als Download bereitgestellt wird. Da kein Webserver zur Verfügung steht und nur mit JavaScript gearbeitet wird, muss das Downloadfile dynamisch erstellt werden. Dazu wird FileSaver.js eingesetzt. FileSaver.js braucht dabei für aktuelle Browser die „Blob Funktionalität“ von HTML5, welche aber noch nicht offiziell standardisiert worden ist, und für ältere Modelle wird eine data:URI erstellt.

### Code Obfuscation

JavaScript Code wird nicht kompiliert und ist daher sehr anfällig gegen Manipulationen. Ein einfacher Text-Editor genügt, um massive Eingriffe in die Logik vorzunehmen. Selbst auf eBook Readern ist man davor nicht sicher, deshalb wird bei SecureExam der JavaScript Code Obfuscated. Code Obfuscation ersetzt alle Strings im Code durch codierte Versionen davon, entfernt jegliche Codestruktur und minimalisiert alle Variablen so weit wie möglich.

|  |  |
| --- | --- |
| **Ohne Obfuscation** | **Mit Obfuscation** |
|  |  |

Tabelle 5: JavaScript Obfuscation

Ohne spezifische Tools zum “De-Obfuscaten” von JavaScript Code, ist das Resultat für Menschen praktisch unlesbar, geschweige denn, in effizienter Form manipulierbar. Zusätzlich kommt dazu, dass die sich die Studenten in einem Prüfungsumfeld befinden und nur begrenzt Zeit haben.

Somit kann davon ausgegangen werden, dass der JavaScript Code während der Prüfung nicht manipuliert werden kann.

# Umsetzung

## Kryptographie

### Verschlüsselung

Die Umsetzung kryptographischer Funktionalitäten in C# ist sehr einfach. Von Hause aus bietet Microsoft eine eigene Library mit fast allen möglichen Algorithmen. Es ist nur folgendes using Statement nötig:



Abbildung 20: Kryptographie Import in C#

Alle Symetrischen Algorithmen der System.Security.Cryptography Library implementieren das IDisposable Interface. Das heisst, man kann Sie bequem in einem Using-Scope verwenden, womit sie nach dem Verschlüsselungsvorgang direkt dem Garbage Collector übergeben werden.



Abbildung 21: AES mit Using-Scopes

Die Verschlüsselten Daten werden als Byte-Array zurückgegeben. Jedoch ist es oft nötig, diese in Strings aus druckbaren Zeichen zu konvertieren. Es ist wichtig, im Vorfeld festzulegen, mit welchem Encoding man arbeitet. SecureExam arbeitet generell mit Hex Codierten Strings, dies aus Grund der einfacheren Ablesbarkeit der realen Werte gegenüber z.B. einem Base64 String.

### Entschlüsselung

Die Entschlüsselung der Fragen und der Prüfungsdaten findet direkt während der Prüfung via JavaScript statt. CryptoJS braucht ein paar spezifische, nicht dokumentierte Anpassungen, um mit eigenen IV’s und Keys zu arbeiten.



Abbildung 22: CryptoJS AES decryption mit custom IV & Key

Zuerst muss ein CipherParams Objekt erzeugt werden. Dies erfolgt auf der Zeile 768. In diesem können nun eigene Einstellungen vorgenommen werden. Wichtig ist, dass hier der key, iv und der ciphertext angegeben wird. Diese müssen bereits zu einem Byte-Array konvertiert worden sein. Anschliessend kann normal die decrypt-Methode von AES aufgerufen werden, jedoch mit dem CipherParams Object anstelle des Ciphertext-Parameters. Auch muss nochmal der Key und mittels „Options Object“ der eigene IV angeben werden.

### Passwort hashing

#### SecureExam.exe

Wie bei AES ist auch SHA256 Teil der System.Security.Cryptography Library und implementiert das IDisposable Interface. Durch den Aufruf von ComputeHash wird der Hash berechnet und als Byte Array zurückgegeben.



Abbildung 23: SHA256 Hashing in C#

Nach dem erstmaligen Generieren des Hashs wird das Hash-Chaining durchgeführt. D.h. die ComputeHash Methode wird noch so oft durchgeführt, wie in der Hashingkonfiguration eingestellt.

#### JavaScript

Hashing in CryptoJS ist sehr einfach. Es steht die Funktion CrytoJS.SHA256() zur Verfügung, welche den Hash berechnet und als Byte-Array zurückgibt.



Abbildung 24: SHA256 Berechnung mit CryptoJS

Wie in C# muss in JS genau das gleiche Chaining angewendet werden, da sonst natürlich nicht der gleiche Hash resultiert.

## Konfigurationsdateien

Zum Einlesen der Konfigurationsdateien, welche beide im XML-Format sind, wird das XML.NET Framework verwendet. Die Einstellungen werden über einen Parser eingelesen und der C#.NET-Struktur übergeben.

Durch die Art der Umsetzung von SecureExam ist es ohne grossen Aufwand möglich, die Applikation so zu erweitern, dass verschiedene Dateitypen zur Konfiguration zusätzlich implementiert werden können.

### Prüfungskonfigurationsdatei

Die Datei zur Angabe von Konfigurationen bezüglich der Prüfung wird über die Konsolenparameter angegeben und muss im XML-Format sein.



Abbildung 25: Beispiel einer Prüfungskonfigurationsdatei

Es existieren 14 verschiedene Parameter, welche in der Bedienungsanleitung beschrieben werden. Ein Teil der Konfigurationen ist optional, während folgende Parameter obligatorisch sind:

* examDate
* startTime
* endTime
* duration
* historyTimeMaxVariance
* internalTimeMaxVariance

Wird einer dieser Parameter nicht gesetzt, erscheint beim Erstellen der Prüfung eine Fehlermeldung, welche die fehlende Konfiguration angibt.

#### Prüfungshinweise

#### Über das Element <examNotes> ist es möglich Hinweise zur erstellten Prüfung anzeigen zu lassen. Durch die Verwendung von HTML kann die Darstellung von diesen Hinweisen beeinflusst werden.



Abbildung 26: Links Prüfungshinweise mit HTML, Rechts die Darstellung im Browser

### SecureExam.xml

Die Datei SecureExam.xml muss diesen Namen tragen und im gleichen Verzeichnis wie die Hauptapplikation SecureExam.exe liegen. Sie enthält Konfigurationen, welche die Applikation verwendet, um Prüfungen zu erstellen.



Abbildung 27: Struktur von SecureExam.xml

Alle Angaben in diesem File sind obligatorisch und müssen Angegeben werden, sonst wird eine Fehlermeldung ausgegeben und die Prüfung wird nicht erstellt. Wie in Abbildung 30 ersichtlich ist, werden sowohl die Anzahl von Zufallszahlen für das Erstellen des Passworts definiert als auch Einstellungen zum Ver- und Entschlüsseln der Prüfung mittels AES und SHA256 vorgenommen. Die Verwendung dieser Kryptografie-Parameter werden im Abschnitt 4.2 Kryptographie und in der Bedienungsanleitung genauer erläutert.

## Import

Über die Konsolenparameter muss eine Datei für den Import angegeben werden. Der Dateityp dieses Files muss ebenfalls angegeben werden, dazu wird ein weiterer Parameter gesetzt. Zum Zeitpunkt der Abgabe sind XML und ODT die zwei möglichen Dateitypen für den Import, jedoch wurde die Applikation so aufgebaut, dass weitere Importmöglichkeiten jederzeit hinzugefügt werden können.

### XML

Der Import über eine XML-Datei erfolgt über einen in C# entwickelten Parser, der das XML Framework von .NET verwendet. Die einzelnen Elemente, welche verwendet werden können, werden vom Parser gesucht und in die C# Instanzen geschrieben. Nach dem Laden der Datei wird sofort geprüft, ob es sich dabei um eine Gültige XML-Datei handelt.



Abbildung 28: XML-Auszug mit einer Frage

Die Struktur erlaubt sowohl optionale als auch obligatorische Einstellungsmöglichkeiten, welche in der Bedienungsanleitung ausführlicher beschrieben werden. Zur Erstellung der Prüfung muss jede Frage in einem <legend>-Tag gestellt werden, für die Antwort stehen Checkboxen und Textfelder zur Auswahl.

### Open Office

Eine weitere implementierte Möglichkeit zum Importieren von Prüfungsdetails ist die Angabe einer Open-Office-ODT-Datei, welche nach bestimmten Regeln aufgebaut sein muss. Um die rohen XML-Daten zu formen wird XSLT verwendet. Im ersten Schritt wird dabei das XML-File aus der komprimierten ODT-Struktur extrahiert und dem XSLT-Parser übergeben. Mithilfe des XSLT-Files, welches in der SecureExam.exe als Ressource vorliegt, wird die ODT File in eine bessere, lesbare Struktur gebracht.



Abbildung 29: Auszug XSLT

Weil die Fragen der Prüfung und die dazugehörigen Antworten zuerst zusammengebracht werden müssen, wird ein zweites XSLT-Dokument benötigt, welches dann ein sauberes, nach den Vorgaben strukturiertes XML-File erstellt.

Die generierte XML-Datei ist nun so weit vorbereitet, dass sie im letzten Schritt dem bereits beschriebenen XML-Parser(vgl. 5.4.1 XML) übergeben werden kann. Nach diesem letzten Arbeitsvorgang sind die Daten im Programm angekommen und der Import ist dadurch abgeschlossen.

### Studenten Import

Die Angaben aller Studenten, welche die Prüfung absolvieren müssen, werden in einer XML-Datei eingetragen und der Applikation übergeben. Der XML-Parser importiert alle Angaben der Studenten und generiert daraus individuelle Anmeldeoptionen.



Abbildung 30: Beispiel von Einträgen im Studenten Import File

## Export

### Prüfungsdatei

Die Prüfung wird über die Exportfunktion in eine HTML-Datei gespeichert, welche CSS zur Darstellung und JavaScript für die Funktionalität verwendet. Dieser Export geschieht über eine so genannte Skeleton-Datei. Während CSS und JavaScript bereits im Skeleton-File integriert sind, müssen die Prüfungsdetails noch generiert werden.



Abbildung 31: Auszug aus der Skeleton-Datei

Das Skeleton-File mit Platzhaltern dient als Vorlage für die Prüfungsdatei. Während dem generieren der Prüfung werden diese Platzhalter durch die Import-Daten ersetzt.



Abbildung 32: Das Ersetzen der Platzhalter als Auszug aus der C#-Exportfunktion

### Studenten Login Daten

Zum Zeitpunkt der Projektabgabe werden die Studenten Login Daten in eine die XML-Datei exportiert, die denselben Namen trägt, wie die exportierte Prüfung. Dieses File ist eine Ansammlung von Einträgen zu jedem Studenten, der für die Prüfung zugelassen wurde.



Abbildung 33: Beispiel von exportierten Studenten Login Daten

Sie beinhaltet die Angaben Name, Vorname, Immatrikulationsnummer und das generierte Passwort, das dem Studenten vor der Prüfung übergeben werden muss.

## Digitale Manipulationsmöglichkeiten

### Zeit

Wie im Konzept festgelegt, wird jede Sekunde die aktuelle Systemzeit ausgelesen und gegen einen Zeitverlauf und eine eigene, interne Uhr geprüft. Die Hauptproblematik bei einer eigenen, in JavaScript realisierten Uhr ist, dass der Update Event nicht auf die Millisekunde genau ausgeführt wird. Das heisst, dass die window.setInterval( function, interval ) Funktion nur begrenzt genau ist.

Um für das Problem eine Lösung zu finden, mussten zuerst die Lag Zeiten Analysiert werden:



Tabelle 6: CPU Lag Analyse (Testsystem: Intel I7-4500U Windows 8.1 64Bit mit Chrome 40.0.2214.28 beta-m (64-bit))

Aus der Analyse kann man folgende Schlüsse ziehen:

1. Ist das Browserfenster mit dem JavaScript Code aktiv, treten Aufrufverzögerungen < 20ms auf.
2. Ist das Browserfenster inaktiv, d.h. der Benutzer surft auf einem anderen Fenster, beträgt die Verzögerung in etwa eine Sekunde.
3. Eine hohe CPU Last bewirkt bei aktivem und inaktivem Fenster eine genauere Ausführung der setIntveral Methode. Wobei dieser Effekt bei inaktivem Fenster ca. jede zweite Sekunde auftritt.
4. Die grössten CPU-Lags treten auf, wenn das Fenster inaktiv ist.

Da es keine Möglichkeit gibt, mittels JavaScript diesen „CPU Lag“ zu beeinflussen, treten folgende Möglichkeiten zur Behandlung des CPU Lags auf:

1. Eine Abweichung von < 20ms pro Sekunde akzeptieren, Browserfenster-Fokuswechsel detektieren und, im Falle eines Wechsels, die Prüfung pausieren. Die komplette Sicherheit bleibt bestehen, der Benutzer muss sich aber nach dem Tab-Wechsel neu authentifizieren.
2. Browserfenster-Fokuswechsel erlauben und die Abweichung auf >2s stellen. Dies hebelt aber das gesamte Sicherheitsprinzip der internen Uhr aus, da mittels geschickter Zeitmanipulation pro Sekunde theoretisch 2 Sekunden gewonnen werden könnten. Die Sicherheit wird auf die Überprüfung des Zeitverlaufes reduziert.

Es wird empfohlen, die Variante a) zu verwenden, da sonst der Sicherheitsmechanismus beträchtlich eingeschränkt und verschlechtert wird. Die betreffenden Einstellungen sind mittels der Konfigurationsdatei „Settings.xml“ anpassbar:



Abbildung 34: Konfigurationsparameter zum Handling des CPU Lags (Parameter in ms)

In der nachfolgenden Abbildung erkennt man die Zeitvergleichsfunktion und der Akzeptanz eines gewissen CPU Lags (variance Variable) sowie die codestelle zur kompensierung des CPU Lags. Es wird davon ausgegangen, dass die Systemzeit valid ist, wenn diese innerhalb der Varianz liegt. Also wird die interne Zeit um die Varianz nachgestellt und läuft wieder genau.



Abbildung 35: DateCompate Funktion und Methode zur Kompensation des CPU Lag

### Internet Zugriff

Mittels JavaScript kann sehr einfach, dynamisch ein neues Bild erzeugt werden. Dieses kann mit EventListeners versehen werden. SecureExam verwendet die Imageevents für OnLoad und OnError.

* OnLoad wird beim erfolgreichen Laden des Bildes aufgerufen, was eine aktive Internetverbindung darstellt.
* OnError wird bei einem Fehler beim Herunterladen des Bildes aufgerufen, ergo die Internetverbindung ist nicht aktiv.

Falls das Bild nicht gefunden wird, wird OnLoad aufgerufen mit 404 ErrorCode. Somit funktioniert die Methode auch, wenn versucht wird, mittels eigenen DNS Einträgen die Funktionalität zu beeinträchtigen.



Abbildung 36: Dynamische Einbindung eines neuen Bildes zur Prüfung der Internetverbindung

Als Bild URL wird das Logo der ZHAW Seite von Karl Rege verwendet. Dies wurde so ausgewählt, um eine lange Lebensdauer der Funktion zu garantieren.

## UI-Design

### Farben

Die Farbwahl wurde aufgrund der ZHAW Farben auf Blau festgelegt. Der Titelbalken des GUIs verwendet dasselbe Blau wie das ZHAW Logo. Die Hintergrundfarbe der Prüfungshinweise wurde aufgrund visueller Trennung ein wenig abgedunkelt und das weiss für die Fragen repräsentiert ein normales Blatt.

### Authentifizierungsansicht



Abbildung 37: Authentifizierungsansicht

Wie im Konzept festgelegt, muss der Student vier Felder ausfüllen. Diese sind Name, Vorname, Immatrikulationsnummer und das Prüfungspasswort. Wichtig ist, dass das Prüfungspasswort aufgrund seines Base64-Encodings Key-Sensitive ist!

### Page-Mode

Der Page-Mode ist für eBook Reader gedacht und zeigt jederzeit nur eine Frage auf einmal an.



Abbildung 38: Prüfung im Page-Mode

Mittels dem Klick auf Zurück bzw. Weiter kann die Frage gewechselt werden. Die Fragen wechseln in einem Endlos-Loop, das heisst dass der Klick auf Weiter bei der letzten Frage den Student wieder zur ersten Frage navigiert. Beim Wechseln zwischen den Fragen werden absichtlich keine Animationen verwendet, da dies aufgrund der Display-Eigenschaften von eBook Readern (eInk) kontraproduktiv für das „Look and Feel“ wäre.

### Scroll-Mode

Der Scroll-Mode wird für die Anzeige mittels Computern oder Tablets mit LCD Bildschirmen verwendet.



Abbildung 39: Prüfung im Scroll-Mode blendet die Hinweise aus

Wie auf der Grafik oberhalb ersichtlich, verschwinden die Prüfungshinweise beim Scrollen nach unten. Dies wurde so designt, da man die Hinweise nur am Anfang der Prüfung liest und anschliessend auf das Lösen der Fragen fokussiert ist. Deshalb wurde der Platz der Prüfungshinweise zugunsten von mehr Platz für Prüfungsfragen freigegeben. Natürlich werden die Hinweise wieder eingeblendet, wenn man zurück zum Start scrollt.

# Testing

## Konzept

Zum Überprüfen der Funktionalität von C# Code, eignen sich Unit-Tests besonders gut. Da von Microsoft bereits eine Testing-Suite angeboten wird, macht es Sinn, diese auch zu verwenden. Wichtig zu beachten ist, dass die Tests nur mit Visual Studio 2012/2013 Ultimate funktionieren. Bei den einfacheren Versionen von Visual Studio ist die Funktionalität nicht enthalten.

Mittels der integrierten Funktion „Analyze Code Coverage“ wird zudem sichergestellt, dass alle Codezeilen von den zu testenden Klassen mit Tests abgedeckt werden.

## Realisierung

### Fakes

Das von Java bekannte Mocking wird in C# mittels Fakes umgesetzt. Um Fakes zu nutzen, muss zuerst von der entsprechenden Referenz ein Fake Assembly erzeugt werden. Dies erfolgt sehr einfach mittels Rechtsklick auf die Referenz / Add Fakes Assembly.

Abbildung 40: Add Fakes Assembly

Dies generiert einen Subnamespace Fakes. Bei SecureExam wäre dies dann SecureExam.Fakes. Nun kann mittels Lambda Ausdrücken die Funktionalität einer Klasse / Funktion simuliert werden. Die Klassen sind immer zusammengesetzt aus „Shim“ und dem Klassennamen.



Abbildung 41: C# Fakes

In der Abbildung oberhalb wird auf der ersten Zeile die Methode export in der Klasse Facade simuliert. Dadurch, dass im Lambda Ausdruck nur {} steht, wird beim Aufruf der Funktion nichts passieren.

Auf der dritten Zeile wird die Funktion readData auch von der Facade-Klasse simuliert. Sie wirft beim Aufruf eine NotImplementedException.

So kann sehr einfach jegliches Verhalten von anderen Klassen simuliert werden.

### Konsolenausgabe abfangen

Oftmals ist es bei Tests für die GUI / Konsolenklasse nötig, den Konsolenoutput einer Applikation abzufangen und zu prüfen. Dies wird in C# so gelöst, dass man die Out Eigenschaft der Konsole überschreibt, um den Output auf einen eigenen StreamWriter umzuleiten.



Abbildung 42: Konsolenausgabe umleiten in C#

## Testresultat

Professor Rege hat während dem Testen darauf hingewiesen, bei 50% Code-Abdeckung aufzuhören, da dies für den Umfang der Projektarbeit mehr als genug sei. Entsprechend behandeln nachfolgende Test-Resultate nur 50% des eigentlichen Quelltextes.

### Unit Tests

Insgesamt wurden 63 Unit Tests erfasst, welche die wichtigsten Funktionen der SecureExam-Applikation abdecken. Nachfolgend ein Screenshot des Test Explorers nach dem erfolgreichen Durchlaufen der Unit Tests.



Abbildung 43: TestExplorer mit den Unit Tests

### Code Coverage Analyzer



Abbildung 44: Auszug der Code Coverage Analyzer Ausgabe

Wie als Ziel festgelegt, wurde eine Code Abdeckung von ca. 50% erreicht. Mit den Tests werden wie auf der Abbildung ersichtlich, 574 von 1153 Codeblocks abgedeckt, was in etwa 52% des gesamten Quelltextes entspricht.

# Resultate

Als Resultate werden Bildschirmaufnahmen von der generierten Prüfung auf verschiedenen Geräten präsentiert.

## Prüfung auf PC mit Scroll-Funktion



Abbildung 45: Prüfung auf PC mit Scroll-Funktion

## Prüfung auf Tablet

### Prüfung im Paging-Mode auf Tablet mit LCD-Bildschirm

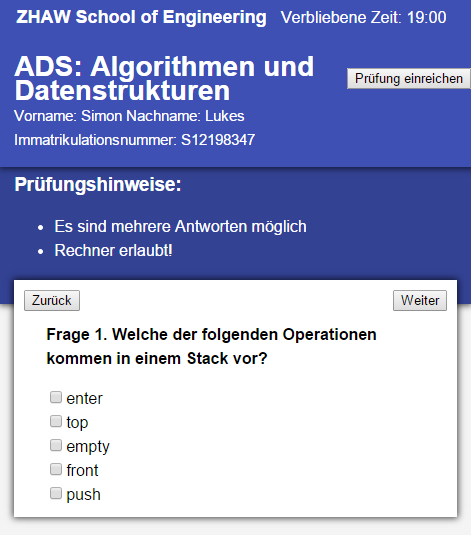


Abbildung 46: Prüfung im Paging-Mode auf Tablet mit LCD-Bildschirm

### Prüfung im Paging-Mode auf Tablet mit eInk Bildschirm

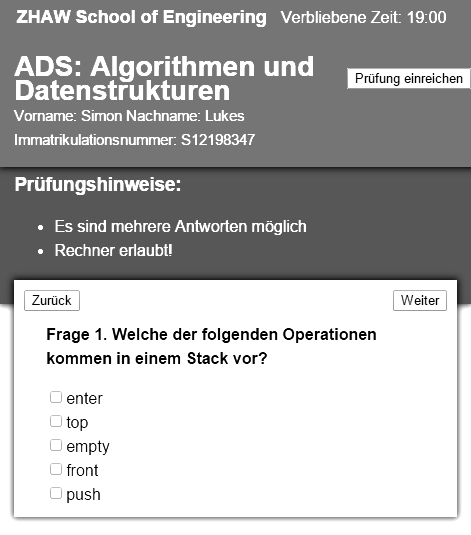


Abbildung 47: Prüfung im Paging-Mode auf Tablet mit eInk Bildschirm

# Diskussion und Ausblick

## Erreichte Ergebnisse

Die Applikation mit allen Funktionen, Import- und Exportmöglichkeiten wurden gemäss Aufgabenstellung oder durch Absprach mit dem Dozenten implementiert. Die Idee eine Prüfung absolvieren zu können, ohne ein Blatt Papier zu verwenden wurde erfolgreich umgesetzt.

Das Ziel war, eine Möglichkeit zu haben, mit der Prüfungen durchgeführt werden können, ohne Druckerkosten und Aufwand mit Papier zu haben. Dieses Ziel wurde mehrheitlich erreicht, jedoch besteht immer noch ein gewisser Aufwand für den Dozenten, die Prüfung einzusammeln und zu korrigieren.

## Ausblick und Chancen

Der ganze Aufbau des Projektes wurde so geplant, dass eine Weiterentwicklung nicht nur möglich, sondern erwünscht ist. Mehrere Parameter sind zu der Zeit der Abgabe noch Optional, da nur eine Variante implementiert wurde. Diese Parameter können sehr einfach um weitere Varianten erweitert werden, wenn die entsprechende Funktionalität implementiert wird. Folgende Funktionalitäten könnten eingebaut werden:

* Import von anderen Dateitypen als XML und ODT für die Prüfungsfragen (z.B. Word)
* Import von Konfigurationsdateien mittels INI-File
* Export der Prüfung in andere Dateitypen (z.B. PDF)
* Export der Studenten Anmeldeinformationen in ein anderes Format

Des Weiteren ist es möglich eine zusätzliche Applikation zu entwickeln, welche die Prüfungen kontrolliert und auswertet. Multiple-Choice Aufgaben eignen sich dafür am besten. Sobald Textbasierte Antworten möglich sind, wird das automatisierte Korrigieren sehr schwierig umzusetzen.

# Verzeichnisse

## Literaturverzeichnis

Bray, T. et al., 2006. *W3cSpec.* [Online]   
Available at: http://www.w3pdf.com/W3cSpec/XML/2/REC-xml11-20060816.pdf  
[Zugriff am 4 Dezember 2014].

Kaliski, B., 2000. *ietf.org.* [Online]   
Available at: https://tools.ietf.org/html/rfc2898#section-5.2  
[Zugriff am 11 12 2014].

NIST, 2014. *csrc.nist.gov.* [Online]   
Available at: http://csrc.nist.gov/publications/drafts/fips-202/fips\_202\_draft.pdf  
[Zugriff am 10 12 2014].

Rennhard, M., 2014. *ISI-Vorlesung-02-SecretKeyCryptography,* Winterthur: ZHAW.

## Glossar

## Abbildungsverzeichnis

[Abbildung 1: AES Block Verschlüsselung 12](#_Toc406067192)

[Abbildung 2: ECB Verschlüsselungsmodus 13](#_Toc406067193)

[Abbildung 3: CBC Verschlüsselungsmodus 13](#_Toc406067194)

[Abbildung 4: RSA Verschlüsselung 13](#_Toc406067195)

[Abbildung 5: Inhalt einer ODT-Datei 15](#_Toc406067196)

[Abbildung 6: Inhalt einer DOCX-Datei 15](#_Toc406067197)

[Abbildung 7: Funktionsweise von XSLT 15](file:///C:\Users\Simon\Documents\ZHAW\Projekt2\Files\Dokumentation\Dokumentation.docx#_Toc406067198)

[Abbildung 8: Systemüberblick 23](#_Toc406067199)

[Abbildung 9:Domänenmodell SecureExam 24](#_Toc406067200)

[Abbildung 10: Beispiel eines Exception rethrowings 24](#_Toc406067201)

[Abbildung 11: Multi-User AES Verschlüsselung 26](#_Toc406067202)

[Abbildung 12: Multi-User AES Entschlüsselung 27](#_Toc406067203)

[Abbildung 13: Hashing Ablauf 28](#_Toc406067204)

[Abbildung 14: Manipulation der internen Uhr detektieren 30](#_Toc406067205)

[Abbildung 15: Vergleich der internen Uhr mit der Systemuhr 31](#_Toc406067206)

[Abbildung 16: UI Design Idee 32](#_Toc406067207)

[Abbildung 17: Login-Skizze 32](#_Toc406067208)

[Abbildung 18: Skizze Paging Mode 33](#_Toc406067209)

[Abbildung 19: Skizze Scroll Mode 33](#_Toc406067210)

[Abbildung 20: Kryptographie Import in C# 36](#_Toc406067211)

[Abbildung 21: AES mit Using-Scopes 36](#_Toc406067212)

[Abbildung 22: CryptoJS AES decryption mit custom IV & Key 37](#_Toc406067213)

[Abbildung 23: SHA256 Hashing in C# 37](#_Toc406067214)

[Abbildung 24: SHA256 Berechnung mit CryptoJS 38](#_Toc406067215)

[Abbildung 25: Beispiel einer Prüfungskonfigurationsdatei 38](#_Toc406067216)

[Abbildung 26: Links Prüfungshinweise mit HTML, Rechts die Darstellung im Browser 39](#_Toc406067217)

[Abbildung 27: Struktur von SecureExam.xml 39](#_Toc406067218)

[Abbildung 28: XML-Auszug mit einer Frage 40](#_Toc406067219)

[Abbildung 29: Auszug XSLT 40](#_Toc406067220)

[Abbildung 30: Beispiel von Einträgen im Studenten Import File 41](#_Toc406067221)

[Abbildung 31: Auszug aus der Skeleton-Datei 41](#_Toc406067222)

[Abbildung 32: Das Ersetzen der Platzhalter als Auszug aus der C#-Exportfunktion 42](#_Toc406067223)

[Abbildung 33: Beispiel von exportierten Studenten Login Daten 42](#_Toc406067224)

[Abbildung 34: Konfigurationsparameter zum Handling des CPU Lags (Parameter in ms) 43](#_Toc406067225)

[Abbildung 35: DateCompate Funktion und Methode zur Kompensation des CPU Lag 44](#_Toc406067226)

[Abbildung 36: Dynamische Einbindung eines neuen Bildes zur Prüfung der Internetverbindung 45](#_Toc406067227)

[Abbildung 37: Authentifizierungsansicht 45](#_Toc406067228)

[Abbildung 38: Prüfung im Page-Mode 46](#_Toc406067229)

[Abbildung 39: Prüfung im Scroll-Mode 46](#_Toc406067230)

[Abbildung 40: Add Fakes Assembly 47](#_Toc406067231)

[Abbildung 41: C# Fakes 47](#_Toc406067232)

[Abbildung 42: Konsolenausgabe umleiten in C# 48](#_Toc406067233)

[Abbildung 43: TestExplorer mit den Unit Tests 48](#_Toc406067234)

[Abbildung 44: Auszug der Code Coverage Analyzer Ausgabe 49](#_Toc406067235)

[Abbildung 43: Beispiel eines Studenten Import Files 70](#_Toc406067236)

## Tabellenverzeichnis

[Tabelle 1: Vergleich AES und RSA 25](#_Toc406067237)

[Tabelle 2: Vergleich SHA3 und PBKDF2 27](#_Toc406067238)

[Tabelle 3: Vergleich ODT und Word 29](#_Toc406067239)

[Tabelle 4: Vergleich HTML und PDF 29](#_Toc406067240)

[Tabelle 5: JavaScript Obfuscation 35](#_Toc406067241)

[Tabelle 6: CPU Lag Analyse (Testsystem: Intel I7-4500U Windows 8.1 64Bit mit Chrome 40.0.2214.28 beta-m (64-bit)) 43](#_Toc406067242)

# Anhang

## Projektmanagement

### Zeitplan



### Sitzungsprotokolle

#### Sitzung 1:

**Datum:** 15. September 2014

**Anwesend:** Karl Rege, Simon Lukes, Daniel Jampen **Abwesend**: -

**Erledigte Arbeiten:**

* keine

**Besprochene Tendenzen:**

* Aufgabenstellung besprochen und eine grobe Übersicht aufgezeichnet.  
    
  Zeitmanipulationsschutz wird als Muss-Funktion festgelegt, da dort die grösste Gefahr gesehen wird.
* Sitzungstermine festgelegt: Wöchentlich Montags um 13:40

**Zu erledigende Arbeiten:**

* Projektplan ausarbeiten (grob)
* Einarbeiten, einlesen, Überblick verschaffen
* Risiken notieren

#### Sitzung2

**Datum:** 22. September 2014

**Anwesend:** Karl Rege, Simon Lukes, Daniel Jampen **Abwesend**: -

**Erledigte Arbeiten:**

* Überblick verschaffen und einlesen
* Zeitplan erstellt
* Risiken notiert

**Besprochene Tendenzen:**

* Word- zu PDF-Formular konvertieren geht nicht
* Colorbox als eBook Reader sieht gut aus ☺
* Verschlüsselungsidee ausgearbeitet => noch mit TeBe anschauen
* Verschlüsselungssicherheit: mind. 1 Tag
* HTML + JS als Technologie ausgewählt

**Zu erledigende Arbeiten:**

* Verschlüsselung besprechen mit Herrn Tellenbach
* Zeitplan anpassen (=> Gerät evaluieren an den Schluss, Input Herrn Rege)

#### Sitzung3

**Datum:** 29. Oktober 2014

**Anwesend:** Karl Rege, Simon Lukes, Daniel Jampen **Abwesend**: -

**Erledigte Arbeiten:**

* Verschlüsselungskonzept erstellt und mit Herrn Tellenbach besprochen
* CryptoJS Machbarkeitstest => Verschlüsselungsidee funktioniert

**Besprochene Tendenzen:**

* Verschlüsselungskonzept:  
  
* Alle Inputfiles in XML
* SecureExam Applikation: Keine GUI, nur Kommandozeile

**Zu erledigende Arbeiten:**

* Klassendiagramm
* SVN Einrichten
* SecureExam Applikation implementieren (beginnen)

#### Sitzung4

**Datum:** **:** 6. Oktober 2014

**Anwesend:** Karl Rege, Simon Lukes, Daniel Jampen **Abwesend**: -

**Erledigte Arbeiten:**

* Klassendiagramm
* SVN Einrichten
* Begonnen mit der Implementation der Applikation

**Besprochene Tendenzen:**

* Word Input als Alternative zu XML => zu komplexes Format. Wird nicht weiter verfolgt
* Fragen einlesen mittels ODT sollte gehen, gute interne XML Struktur  
  Idee: ODT mittels XSLT in unser XML Format konvertieren, dann normal parsen
* Student ID auf Vorname + Nachname + Immatrikulationsnummer + Random Chars festgelegt

**Zu erledigende Arbeiten:**

* Domänenmodell erstellen
* Applikation weiterprogrammieren
* Eigenes XML Format für Questionfile definieren

#### Sitzung5

**Datum:** 13. Oktober 2014

**Anwesend:** Karl Rege, , Daniel Jampen **Abwesend**: Simon Lukes

**Erledigte Arbeiten:**

* SecureExam in Alpha 1 ☺
* Domänendiagramm erstellt
* Format für Questionfile (XML) definiert

**Besprochene Tendenzen:**

* SecureExam Alpha 1 Problem mit PBKDF2:  
  PBKDF2 Implementationen von C# und CryptoJS sind unterschiedlich => andere Resultate  
  => wir nehmen SHA-3, sowieso bessser da anpassbarer für eBook Reader (performance)
* XML Parser für Questionfile implementiert, sieht gut aus. Next: XSLT für ODT Import erstellen

**Zu erledigende Arbeiten:**

* SHA 3 anstelle von PBKDF2 implementieren
* ODT Import mittels XSLT und bestehendem XML Parser
* Save / Export Funktion überlegen. Wie? localDB? SaveAs im Browser? Verschlüsseln?

#### Sitzung6

**Datum:** 20. Oktober 2014

**Anwesend:** Karl Rege, Simon Lukes, Daniel Jampen **Abwesend**: -

**Erledigte Arbeiten:**

* SHA-3 Implementiert. Funktioniert super. Iterationen konfigurierbar
* AES Ver- und Entschlüsselung funktioniert => SecureExam Alpha 2
* Save Funktion, Möglichkeiten angeschaut

**Besprochene Tendenzen:**

* ODT Import: XSLT Problem, es braucht Format Richtlinien. XSLT ist zum Teil sehr speziell, da wir noch nie XSLT verwendet haben, schick uns Herrn Rege ein Beispielfile
* Save Funktion mittels FileSaver.js

**Zu erledigende Arbeiten:**

* Weiter implementieren
* Save Funktion umsetzen
* Zeitmanipulationsschutz

#### Sitzung7

**Datum:** 27. November 2014

**Anwesend:** Karl Rege, Simon Lukes, Daniel Jampen **Abwesend**: -

**Erledigte Arbeiten:**

* ODT Import, XSLT works. XML aus ODT extrahieren muss noch implementiert werden
* Save Funktion implementiert.
* Design idee: Material Design

**Besprochene Tendenzen:**

* Save Funktion: Export mit userSecret verschlüsseln
* UI Design OK
* Dokumentation: Vorgabe? < 50 Seiten

**Zu erledigende Arbeiten:**

* Zeitmanipulationsschutz gemäss Konzept
* Abklären ob eBook Reader HTML Files von der SD-Karte lesen können
* Design anpassen: Name, Vorname in Titelliste
* Neue Ideen zum Umsetzen:
  + Prüfungsfenster
  + Öffnungszeit und Anzahl Öffnungen loggen und in Export schreiben
  + AutoSave Funktion
  + TabFocus lost
  + InternetConnection check

#### Sitzung8

**Datum:** 3. November 2014

**Anwesend:** Karl Rege, Simon Lukes, Daniel Jampen **Abwesend**: -

**Erledigte Arbeiten:**

* ODT Import fertig
* Export Ergänzungen (Log, Userdetail etc.)
* Design anpassungen gemäss Sitzung 7
* SecureTime implementiert
* SecureExamSettings Klasse als „Session“ serialisiert in LocalDB
* AutoSave und resume
* Prüfugszeitfenster => konfigurierbar via Settings.xml
* Internetverbindungscheck
* Timeleft wird berechnet und angezeigt. Prüfung wird nach Ablauf beendet.
* JS Librarys in Skelton inline => html File nun 48 kb gross

**Besprochene Tendenzen:**

* Prfüngszeitraum und Prüfungsdauer sollten mit Fragen verschlüsselt werden. Manipulationsschutz
* Feature Freeze nächste Woche

**Zu erledigende Arbeiten:**

* JS Code Refactoring
* Testing des C# Codes
* GUI Login => vier Felder
* Tolino’s testen
* Settings.xml soll auch Prüfungshinweise beinhalten und Security Features konfigurieren

#### Sitzung9

**Datum:** 10. November 2014

**Anwesend:** Karl Rege, Simon Lukes, Daniel Jampen **Abwesend**: -

**Erledigte Arbeiten:**

* GUI Login mit vier Feldern
* JS Refactoring
* Settings.xml angepasst
* Neues Parameterfile für Applikation erstellt
* Tolino getestet

**Besprochene Tendenzen:**

* Es ist mühsam Prüfungen auf eBook Reader durchzuführen. Handling laggt => Browser laggt, nicht unsere Schuld

**Zu erledigende Arbeiten:**

* Security-Features ein- und auschalten
* Tests schreiben (C# und JS)
* Code Refactoring C#
* eBook Reader Export in LocalDB implementieren => zweite HTML Seite zum Auslesen der DB erstellen
* Prüfen ob Tolino ein E-Mail Client besitzt.
* Dokumentieren

#### Sitzung10

**Datum:** 17. November 2014

**Anwesend:** Karl Rege, Simon Lukes, Daniel Jampen **Abwesend**: -

**Erledigte Arbeiten:**

* Security-Features ein- und auschaltbar via Settings.xml
* eBook Reder Export done
* Testing bei 55% Code Coverage
* Alter Tolino hat kein E-Mail Client

**Besprochene Tendenzen:**

* Page-Mode für eBook Reader implementieren
* Testing reicht! Stopp by Rege. 50% Code Coverage reichen!

**Zu erledigende Arbeiten:**

* JS Testen und Code Refactoring
* Dokumentation. Nächste Woche Inhaltsverzeichnis besprechen

eTesting Vorteil / Nachteil zeigen, unsere Überlegungen einfliessen lassen.

#### Sitzung11

**Datum:** 24. November 2014

**Anwesend:** Karl Rege, Simon Lukes, Daniel Jampen **Abwesend**: -

**Erledigte Arbeiten:**

* JS Testen und Code Refactoring
* Inhaltsverzeichnis

**Besprochene Tendenzen:**

* Abstract in English + Zusammenfassung in Deutsch am Anfang der Dokumentation

**Zu erledigende Arbeiten:**

* Dokumentieren
* Page-Mode Implementieren

#### Sitzung12

**Datum:** 1. Dezember 2014

**Anwesend:** Karl Rege, Simon Lukes, Daniel Jampen **Abwesend**: -

**Erledigte Arbeiten:**

* Page-Mode implementiert
* Dokumentieren
* Bugfixes

**Besprochene Tendenzen:**

* Inhaltsverzeichnis besprochen

**Zu erledigende Arbeiten:**

* Dokumentieren

#### Sitzung13

**Datum:** 8. Dezember 2014

**Anwesend:** Karl Rege, Simon Lukes, Daniel Jampen **Abwesend**: -

**Erledigte Arbeiten:**

* Dokumentieren

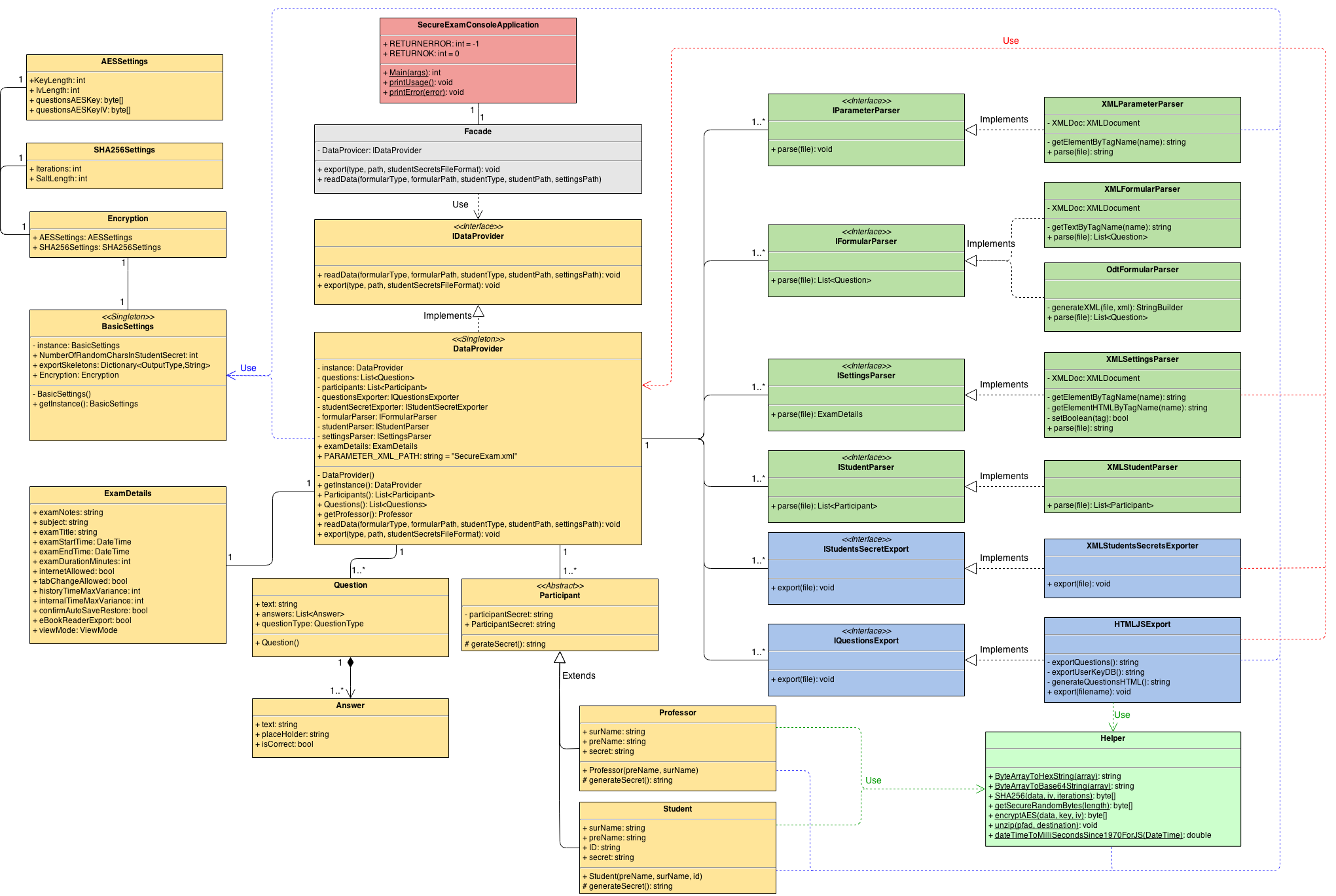
**Besprochene Tendenzen:**

* Finale Fragen zu Dokumentation geklärt
* Letzte Sitzung abgesagt, da unnötig.

**Zu erledigende Arbeiten:**

* Doku abschliessen

### C# Klassendiagramm

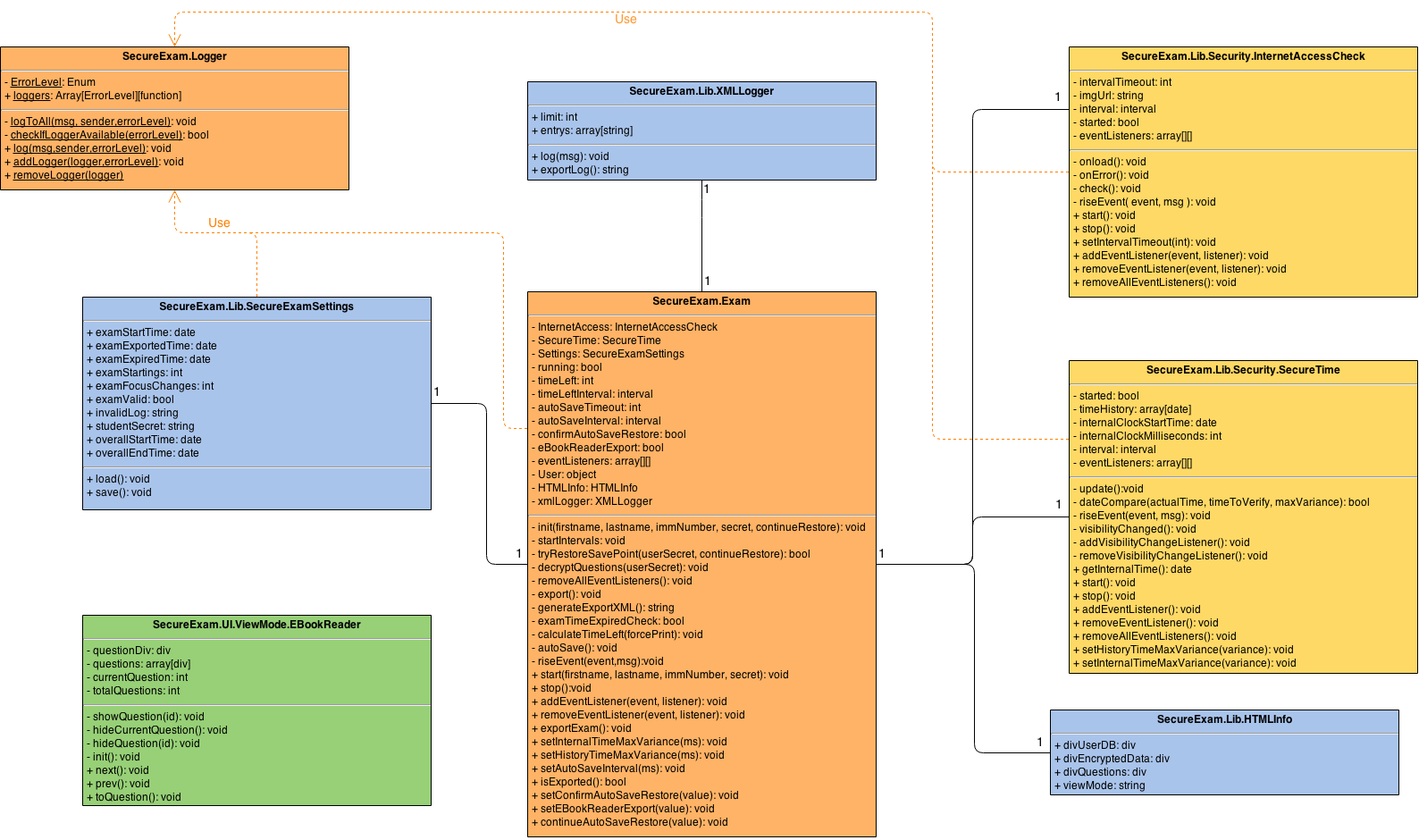


### SecureExam JavaScript Library

#### Overview



#### Klassendiagramm



## Bedienungsanleitung

### Konsolenapplikation

#### Überblick



Die Applikation SecureExam.exe ist eine Konsolenapplikation, welche mehrere Parameter verlangt und allfällige Fehlermeldungen in der Konsole ausgibt.

Eine mögliche Eingabe ist:

secureExam -q questionFile -s studentsFile -o outputFile -p settingsFile

#### Konsolenparameter

Wenn eine Prüfung erstellt werden soll, so müssen folgende Parameter der Konsolenapplikation übergeben werden:

* -q questionFile
* -s studentsFile
* -o outputFile
* -p settingsFile

Diese Parameter sind optional und müssen nur in speziellen Fällen angegeben werden:

* [-qType QuestionFileType]
* [-sType StudentsFileType]
* [-oType OutputFileType]
* [-oStudentSecretsFileFormat studentSecretsFileFormat]

#### Import

Wie in der Übersicht dargestellt, werden vier Dateien benötigt, welche der Applikation übergeben werden, um sie zu einer Prüfungsdatei zu exportierten.

##### SecureExam.xml

Diese Datei wird benötigt, um Konfigurationen für die Applikation vorzunehmen. Sie muss SecureExam.xml heissen und sie muss im gleichen Verzeichnis wie SecureExam.exe abgelegt sein.



Alle Elemente in diesem File sind obligatorisch und müssen angegeben werden.

###### NumberOfRandomCharsInStudentSecret

Dieses Element dient dazu, die Zufallszahlen, welche für das Passwort benötigt werden zu definieren.

###### AESSettings

Die Einstellung <keyLength> definiert die Länge des Schlüssels, welcher für die Ver- und Entschlüsselung verwendet wird. <ivLength> definiert die Länge des Initialisierungsvektors.

###### SHA256Settings

Mithilfe von <iterations> wird die Anzahl von Hash-Durchgängen mit SHA256 festgelegt. Je grösser die Anzahl Iterationen ist, des länger dauert der Anmeldevorgang. Im Gegenzug brauchen aber auch Angreifer länger, um Passwörter durchzutesten. Mit <saltLength> wird die Länge des Salts gesetzt, der für das Hashing verwendet wird.

##### Prüfungseinstellungen.xml

Diese Datei dient zur Konfiguration von Prüfungsdetails. Ihr Name kann frei gewählt werden, muss allerdings mit dem Parameter –p übergeben werden.



Es gibt 14 verschiedene Elemente, welche zum Teil obligatorisch und zum Teil optional sind. In der folgenden Tabelle werden alle Elemente beschrieben:

|  |  |  |
| --- | --- | --- |
| Name | Funktion | Oblig. |
| <subject> | Namen des Faches, wird in die Prüfung geschrieben | Nein |
| <examTitle> | Titel der Prüfung, wird in die Prüfung geschrieben | Nein |
| <examNotes> | Hinweise und Notizen zur Prüfung, wird in die Prüfung geschrieben | Nein |
| <examDate> | Setzt das Datum an dem die Prüfung durchgeführt werden kann | Ja |
| <startTime> | Setzt die früheste Startzeit, vorher kann keine Prüfung gestartet werden | Ja |
| <endTime> | Setzt die späteste Abgabezeit, danach wird die Prüfung automatisch beendet | Ja |
| <duration> | Definiert die Dauer der Prüfung in Minuten | Ja |
| <internetallowed> | Definiert, ob die Verwendung des Internets während der Prüfung erlaubt ist. Mögliche Antworten sind „true“ oder „false“. Als Standard ist „false“ definiert. | Nein |
| <tabChangeAllowed> | Definiert, ob der Tab des Browsers gewechselt werden darf während der Prüfung. Mögliche Antworten sind „true“ oder „false“. Als Standard ist „false“ definiert. | Nein |
| <historyTimeMaxVariance> | Setzt die maximale Abweichung der letzten zehn Zeitmessungen. | Ja |
| <internalTimeMaxVariance> | Setzt die maximale Abweichung der eigenen Zeitrechnung zur Uhr des Systems. Damit wird festgestellt, ob die Systemzeit versucht wird zu manipulieren. | Ja |
| <confirmAutosaveRestore> | Definiert, ob die automatisch gesicherten Daten wiederhergestellt werden dürfen. Mögliche Antworten sind „true“ oder „false“. Als Standard ist „false“ definiert. | Nein |
| <ebookreaderExport> | Definiert, ob der Export der Prüfung als Datei gespeichert wird, oder ob der Export verschlüsselt auf dem Gerät ausgegeben wird. Mögliche Antworten sind „true“ oder „false“. Als Standard ist „false“ definiert. | Nein |
| <viewMode> | Definiert die Ansicht  Mögliche Ansichten sind „scroll“ oder „page“. Als Standard ist „scroll“ definiert.  Bei „scroll“ werden alle Fragen untereinander dargestellt, bei „page“ wird nur immer eine Frage angezeigt und man kann zwischen den Fragen wechseln. | Nein |

##### Studenten.xml

Diese Datei enthält alle Studenten, welche für die Prüfung zugelassen sind, mit ihren Namen, Vornamen und Matrikelnummer. Der Professor muss sich selber ebenfalls eintragen. Aus den Angaben des Files werden alle Logins für die Studenten generiert. Der Dateiname kann frei gewählt werden, muss allerdings mit dem Parameter –s übergeben werden. Der Parameter –sType kann für diese Art auf XML gesetzt werden, da dies allerdings bereits die Standardeinstellung ist, kann dieser Parameter auch weggelassen werden.



Abbildung 43: Beispiel eines Studenten Import Files

##### Prüfungsfragen

Alle Fragen, welche zu einer Prüfung gehören, sind in einer Datei definiert. Es gibt zwei verschiedene Arten von Files, die erstellt werden können. Diesem Absatz folgen die Erklärungen zu den beiden Dateisorten, welche verwendet werden können. Der Dateiname kann frei gewählt werden, muss allerdings mit dem Parameter -q übergeben werden.

###### XML

Eine Methode, die Prüfungsfragen zu erfassen, kann mit einer XML-Datei erfolgen, welche eine bestimmte Struktur haben muss. Der Parameter –qType kann für diese Art auf XML gesetzt werden, da dies allerdings bereits die Standardeinstellung ist, kann dieser Parameter auch weggelassen werden.



Für jede Frag wird ein <question>-Element angelegt, welche mit den Einzelheiten der Frage bestückt wird. Das Element <legend> dient als Behälter für die Formulierung der Frage, hier steht also der Text, welcher auch auf der Prüfung zu sehen sein wird. Als Antwort dienen entweder Checkboxen oder Textboxen, beide werden mithilfe des <input>-Tags eingefügt.

<input type=“checkbox“ value=“enter“ />

Das Attribut type hat entweder den Wert checkbox oder textbox, um das gewünschte Element zu erhalten. Der Wert, welcher angezeigt werden soll, wird mit dem Attribut value gesetzt. Wird bei einer Textbox das Attribut value gesetzt, so wird der definierte Text in der Textbox dargestellt. Des Weiteren ist es möglich einer Textbox ein Attribut placeholder hinzuzufügen. Der Wert dieses Attributs wird dann in grau dargestellt.

###### OpenOffice

Die zweite Möglichkeit, die Prüfungsfragen zu erfassen, kann mit einer ODT-Datei erfolgen. Der Parameter –qType muss für diesen Fall auf ODT gesetzt werden, ansonsten wird eine Fehlermeldung ausgegeben.

Innerhalb des Open Office Dokuments können die Steuerelemente Checkbox und Textarea als Antwort verwendet werden. Jede Frage, die in der Prüfung stehen soll, muss im Open Office Dokument mit „Frage“ beginnen und muss in der Standard Formatvorlage formatiert sein.

#### Export

Der Parameter –o gibt den Speicherort der generierten Prüfung an. Der Dateiname kann frei gewählt werden und der Parameter –oType kann für diese Art auf HTMLJS gesetzt werden, da dies allerdings bereits die Standardeinstellung ist, kann dieser Parameter auch weggelassen werden.

Die Studenten Logins mit den Passwörtern werden in einer Datei gespeichert, welche im selben Verzeichnis wie die Prüfungsdatei liegt und auch denselben Dateinamen trägt, nur die Endung ist unterschiedlich. Dieses Output Format der Logins kann mit dem Parameter –oStudentSecretsFileFormat gesetzt werden, da zu der Zeit der Abgabe der Applikation allerdings nur das Format XML implementiert wurde, kann dieser Parameter weggelassen werden.

### Generierte Prüfung

In dieser Anleitung wird gezeigt, wie das Ausfüllen einer Prüfung abläuft. Die Oberfläche ist einfach gehalten und lässt sich intuitiv bedienen.

#### Startseite

Gleich nach dem Starten des Programms wird folgende Seite angezeigt.

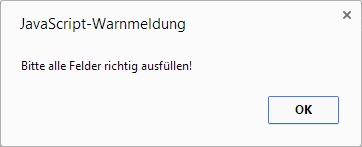


#### Anmelden

Benutzen Sie die Anmeldemaske, um sich am System anzumelden. Alle Angaben stehen in der exportierten XML-Datei, die denselben Dateinamen trägt, wie das HTML-Dokument.



Nach dem Bestätigen der Eingabe durch einen Klick auf „Prüfung starten“, wird die Prüfung entschlüsselt und angezeigt. Falls die Anmeldung nicht erfolgreich war, wird folgende Meldung angezeigt:



Wenn die Anmeldung erfolgreich war, wird nach einer kurzen Wartezeit (von System abhängig) die Prüfung angezeigt.

#### Prüfungsfenster



Auf der Prüfung werden alle Details angezeigt, welche in der Konfiguration definiert wurden. Die Fragen werden gemäss Fragenimport angezeigt und der Name, sowie die Immatrikulationsnummer des Studenten werden unter dem Titel eingeblendet.

In der rechten oberen Ecke sind die verbliebene Zeit und ein Button, mit dem die Prüfung abgegeben werden kann, zu sehen.

#### Beenden

Wird auf den Button „Prüfung einreichen“ geklickt, so erscheint die folgende Meldung, die bestätigt werden muss, um die Prüfung abzugeben.

Nach dem Bestätigen dieser Meldung oder nach dem die Zeit abgelaufen ist, wird die Prüfung beendet und folgende Seite angezeigt:



## Weiteres

### CD mit dem vollständigen Bericht als pdf-File, dem SourceCode und den generierten Code Dokumentationen