



**School of
Engineering**

InIT Institut für angewandte
Informationstechnologie

Projektarbeit (Informatik)

Sichere eTest auf eBook Readern

Autoren

Simon Lukes
Daniel Jampen

Hauptbetreuung

Karl Rege

Datum

16.12.2014

Erklärung betreffend das selbständige Verfassen einer Projektarbeit an der School of Engineering

Mit der Abgabe dieser Projektarbeit versichert der/die Studierende, dass er/sie die Arbeit selbständig und ohne fremde Hilfe verfasst hat. (Bei Gruppenarbeiten gelten die Leistungen der übrigen Gruppenmitglieder nicht als fremde Hilfe.)

Der/die unterzeichnende Studierende erklärt, dass alle zitierten Quellen (auch Internetseiten) im Text oder Anhang korrekt nachgewiesen sind, d.h. dass die Projektarbeit keine Plagiate enthält, also keine Teile, die teilweise oder vollständig aus einem fremden Text oder einer fremden Arbeit unter Vorgabe der eigenen Urheberschaft bzw. ohne Quellenangabe übernommen worden sind.

Bei Verfehlungen aller Art treten die Paragraphen 39 und 40 (Unredlichkeit und Verfahren bei Unredlichkeit) der ZHAW Prüfungsordnung sowie die Bestimmungen der Disziplinar massnahmen der Hochschulordnung in Kraft.

Ort, Datum:

Unterschriften:

.....

.....

.....

.....

Das Original dieses Formulars ist bei der ZHAW-Version aller abgegebenen Projektarbeiten zu Beginn der Dokumentation nach dem Titelblatt mit Original-Unterschriften und -Datum (keine Kopie) einzufügen.

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. Jedoch lieferten diese bis anhin keine zufriedenstellenden Resultate. Teile der Grundproblematiken von eTests waren bei allen Versuchen auszumachen: Manipulationen, unzuverlässige Infrastruktur oder prüfungsinterne Kommunikation beeinträchtigen die Verlässlichkeit der Prüfungsergebnisse.

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 Manipulationsschutzmechanismen aktiv, welche alle Aktionen während der Gesamtdauer des Exams zur Nachverfolgung aufzeichnen und Betrugsversuche detektieren. Durch das dezentrale System von SecureExam wird der Infrastruktureinfluss minimiert 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

Writing exams on paper is still the main choice when there is the need of secure and controllable procedure. A lot of people tried to develop new solutions to accomplish exams on an electronic basis. However, there has not been a lot of significant progress. Parts of the basic difficulties were found in nearly all experiments: manipulation, unreliable infrastructure or communication between the participants interfere the reliability of the results.

This paper demonstrates how these basic difficulties could be addressed and prevented using several new approaches. SecureExam has been developed to provide a platform for creating and realizing electronic tests in a secure and dependable way. The application generates a single, encrypted HTML file from given questions and a list of participants. This HTML website provides a platform independent solution for exam scenarios. For identification and authorization reasons, each participant has to enter his own, randomly generated password along with his personal data in the beginning of the exam. After that, different transparent methods of detecting manipulations are activated in the background. All actions made by the participants are analyzed and logged for later inspection and tracking down manipulations. Due to the decentral architecture of SecureExam, the unreliability problem of IT infrastructures can be minimalized. There is no need for external data sources and therefore it is possible to diminish the risk of internal communication completely by cleverly chosen hardware. With the selected methods manipulations cannot be prevented in general however they are detected and logged by SecureExam. Combining the solutions developed in this study will guarantee a secure, reliable and controllable environment for future electronic exams.

Inhaltsverzeichnis

1	Einleitung.....	9
1.1	Ausgangslage	9
1.1.1	Grundproblematiken eTests.....	9
1.2	Aufgabenstellung.....	10
1.3	Anforderungen	10
1.3.1	Muss Funktionen	10
1.3.2	Wunsch Funktionen.....	11
1.4	Zielsetzung.....	12
1.5	Zielpublikum	12
1.5.1	Prüfungsgenerator.....	12
1.5.2	Prüfung	12
2	Grundlagen	13
2.1	Kryptographie	13
2.1.1	Verschlüsselung.....	13
2.1.2	Passwort Hashing	14
2.2	Dateiformate	15
2.2.1	INI	15
2.2.2	XML.....	15
2.2.3	ODT: Open Office Writer	15
2.2.4	DOCX: Microsoft Office Word	16
2.2.5	XSLT	16
2.2.6	PDF.....	16
2.2.7	HTML	16
2.3	Digitale Manipulationsmöglichkeiten	17
2.3.1	Korrekte Antworten auslesen	17
2.3.2	Zeit.....	17
2.3.3	Internet Zugriff	17
2.3.4	Prüfungsinterne Kommunikation	17
2.4	eBook Reader	18
2.4.1	Einschränkungen	18
3	Analyse	19
3.1	Anwendungsfälle	19
3.1.1	Übersicht	19

3.1.2	Akteure	19
3.1.3	AF01: Prüfung erstellen	19
3.1.4	AF02 Prüfung absolvieren	21
3.2	Nichtfunktionale Anforderungen	22
3.2.1	NFA01 Prüfung ist einfach bedienbar.....	22
3.2.2	NFA02 Export erfolgt transparent für den Probanden.....	22
3.2.3	NFA03 Anmeldevorgang dauert weniger als drei Sekunden.....	22
4	Konzept.....	23
4.1	Systemüberblick	23
4.1.1	Domänenmodell.....	23
4.2	C# Errorhandling.....	24
4.3	Codedokumentation.....	24
4.3.1	C#.....	24
4.3.2	JavaScript.....	24
4.4	Kryptographie.....	24
4.4.1	Verschlüsselung.....	24
4.4.2	Passwort hashing.....	26
4.5	Konfigurationsdateien	27
4.5.1	Entscheid	27
4.5.2	Konzept.....	27
4.6	Import.....	27
4.6.1	Evaluationsmatrix.....	27
4.6.2	Entscheid	27
4.6.3	Konzept.....	28
4.7	Export	28
4.7.1	Evaluationsmatrix.....	28
4.7.2	Entscheid	28
4.7.3	Konzept.....	28
4.8	Digitale Manipulationsmöglichkeiten	28
4.8.1	Zeit.....	28
4.8.2	Internet Zugriff	29
4.8.3	Prüfungsinterne Kommunikation	29
4.9	UI-Design	30
4.9.1	Idee.....	30

4.9.2	Login	31
4.9.3	Prüfung	31
4.10	JavaScript	32
4.10.1	CryptoJS Library	32
4.10.2	FileSaver.js Library	32
4.10.3	Code Obfuscation	32
5	Umsetzung	34
5.1	Kryptographie	34
5.1.1	Verschlüsselung	34
5.1.2	Entschlüsselung	35
5.1.3	Passwort hashing	35
5.2	Konfigurationsdateien	36
5.2.1	Prüfungskonfigurationsdatei	36
5.2.2	SecureExam.xml	37
5.3	Import	37
5.3.1	XML	37
5.3.2	Open Office	38
5.3.3	Studenten Import	39
5.4	Export	39
5.4.1	Prüfungsdatei	39
5.4.2	Studenten Login Daten	40
5.5	Digitale Manipulationsmöglichkeiten	40
5.5.1	Zeit	40
5.5.2	Internet Zugriff	42
5.6	UI-Design	42
5.6.1	Farben	42
5.6.2	Authentifizierungsansicht	42
5.6.3	Page-Mode	43
5.6.4	Scroll-Mode	43
6	Testing	44
6.1	Konzept	44
6.2	Realisierung	44
6.2.1	Fakes	44
6.2.2	Konsolenausgabe abfangen	45

6.3	Testresultat.....	45
6.3.1	Unit Tests.....	45
6.3.2	Code Coverage Analyzer.....	46
7	Resultate.....	46
7.1	Prüfung auf PC mit Scroll-Funktion	46
7.2	Prüfung auf portablen Geräten	47
7.2.1	Prüfung im Paging-Mode auf Tablet mit eInk Bildschirm.....	47
7.2.2	Prüfung im Paging-Mode auf Tablet mit LCD-Bildschirm.....	48
8	Diskussion und Ausblick	49
8.1	Erreichte Ergebnisse	49
8.2	Ausblick und Chancen	49
9	Verzeichnisse	50
9.1	Literaturverzeichnis	50
9.2	Glossar	50
9.3	Abbildungsverzeichnis	51
9.4	Tabellenverzeichnis	52
10	Anhang.....	53
10.1	Projektmanagement.....	53
10.1.1	Zeitplan.....	53
10.1.2	Sitzungsprotokolle	54
10.1.3	C# Klassendiagramm	65
10.1.4	SecureExam JavaScript Library	66
10.2	Bedienungsanleitung.....	68
10.2.1	Konsolenapplikation	68
10.2.2	Generierte Prüfung.....	73
10.3	Weiteres	75
10.3.1	CD mit dem vollständigen Bericht als pdf-File, dem SourceCode und den generierten Code Dokumentationen	75

1 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 Checkboxes mehr und Korrekturen an falsch geschriebenen Wörtern/Antworten können von den Studenten während der Prüfung sauber gemacht werden. Weiter könnten zum Beispiel Programmierprüfungen in gewohntem Umfeld (über die Tastatur geschriebener Sourcecode) abgehalten werden. Noch viele weitere solcher Verbesserungsmöglichkeiten wären möglich und wären sehr einfach mit digitalen Prüfungen realisierbar.

Jedoch gibt es leider 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 ePrüfungen sicher und ohne Bedenken abgehalten werden können.

1.1 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 erledigt werden kann.

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

1.1.1 Grundproblematiken eTests

1.1.1.1 Manipulationen:

In der digitalen Welt gibt es keine absolute Sicherheit. Entsprechend sind Manipulationen in verschiedenen Formen generell und immer möglich. Ohne entsprechende Schutzmechanismen ist man diesen vollkommen chancenlos ausgeliefert.

1.1.1.2 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 (z.B. USB Stick liegen lassen).

1.1.1.3 Unzuverlässige Infrastruktur:

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

1.1.1.4 Gerätemissbrauch zur Kommunikation:

Auch wenn ein in sich abgeschlossenes Netzwerk verwendet wird, sind Notebooks oder Tablets immer noch kommunikationsfähig. Durch geschickten Einsatz von Bluetooth oder WLAN Ad-Hoc Netzwerken wäre eine Kommunikation zwischen den Absolventen möglich.

1.2 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.

1.3 Anforderungen

1.3.1 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 des selber geschriebenen Programms generiert wird, besteht genau aus einer Datei.
Prüfung läuft auf eBook Reader	Die Prüfung muss auf einem eBook Reader durchführbar sein. Die geringere 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 transparent entschlüsseln	Nach der Authentifizierung des Prüfungskandidaten, werden die Fragen mittels einem, pro Student individuellen, Key transparent entschlüsselt.
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.

1.3.2 Wunsch Funktionen

Funktion	Beschreibung
Internetverbindung detektieren	Die Prüfung kann detektieren, 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 Prüfungszeitfenster 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 Verlauf erstellt, welcher festhält, wie oft die Prüfung gestartet wurde und ob Manipulationsversuche auftraten.
AutoSave Funktion	Um bei einem Absturz des Gerätes keine Daten zu verlieren, soll der aktuelle Zwischenstand der Prüfung periodisch gesichert werden.
Sicherheitsfeatures konfigurierbar	Die verschiedenen Sicherheitsfeatures sollen mittels einer Konfigurationsdatei individuell ein- und ausgeschaltet werden können.
Browserfensterwechsel detektieren	Es soll festgestellt werden können, wenn ein Prüfungskandidat auf ein anderes Browserfenster wechselt.
Page-Mode für eBook Reader	Scrollen ist auf eBook Readern träge. Ein Page-Mode entwickelt werden soll. In diesem Modus wird immer nur eine Frage angezeigt und man kann mittels „Weiter“ und „Zurück“ die Frage wechseln.
Parametrisierbare Verschlüsselungsstärke	Die Stärke der Verschlüsselung bzw. die Länge des generierten Passworts kann mittels eines Parameters oder einer Konfigurationsdatei definiert werden.

1.4 Zielsetzung

Ziel der Arbeit ist es, ein Programm zu entwickeln, welches eine verschlüsselte Prüfung generiert. Die Prüfung wiederum besteht aus einer Datei, welche auf einem eBook Reader geöffnet werden kann. Bei dem Start der Prüfung muss sich der Student mit seinem 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 der Abgabe nicht mehr manipuliert werden kann und der Dozent muss die Prüfungsdaten auf eine einfache Art und Weise von dem eBook Reader extrahieren können.

1.5 Zielpublikum

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

1.5.1 Prüfungsgenerator

Die Applikation wird 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.

1.5.2 Prüfung

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

2 Grundlagen

2.1 Kryptographie

2.1.1 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 und die Public Key Verschlüsselung. Anschliessend wird jeweils der aktuelle Standard beschrieben.

2.1.1.1 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 256 Byte), welche letztendlich Stück für Stück verschlüsselt werden.

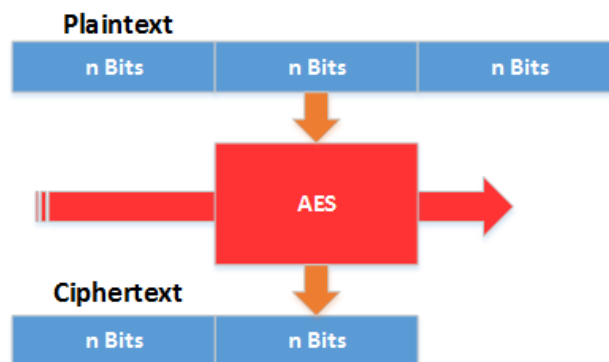


Abbildung 1: AES Block Verschlüsselung

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

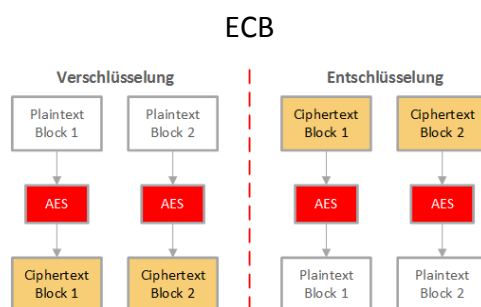


Abbildung 2: ECB Verschlüsselungsmodus

Jeder Block wird unabhängig betrachtet und mit dem gleichen Schlüssel verschlüsselt. Der ECB Mode benötigt per Definition keinen IV oder ähnliches.

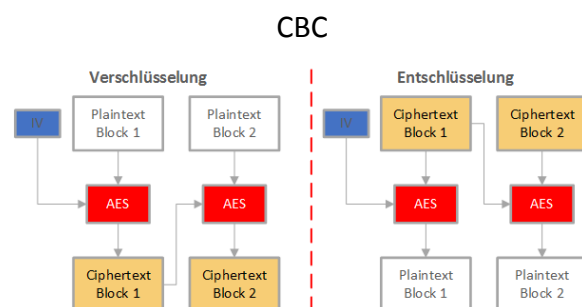


Abbildung 3: CBC Verschlüsselungsmodus

CBC verschlüsselt den ersten Block mit einem Initialisierungsvektor und die folgenden Blöcke mit einem Teil des Ciphertexts des jeweils vorhergehenden Blocks.

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

2.1.1.2 RSA

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

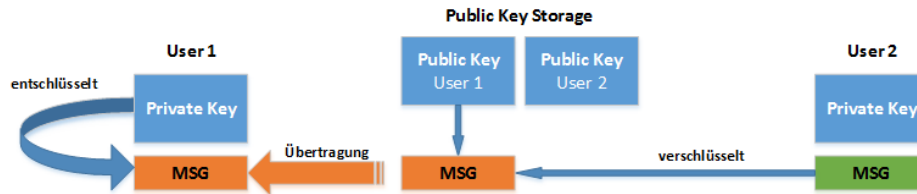


Abbildung 4: RSA Verschlüsselung

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)

2.1.2 Passwort Hashing

Kryptographische Hashfunktionen berechnen aus einem Input variabler Längen einen Output definierter Länge, welcher als Hash bezeichnet wird. 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 Ursprungsnachricht zu ermitteln.
- Collision free: Es muss praktisch unmöglich sein, zwei Nachrichten zu finden, welche denselben 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 Hashfunktion mehrmals nacheinander ausgeführt wird, um die Berechnungszeit des Hashs zu erhöhen. Ein Brute-Force Angriff auf den Hash dauert, je mehr Iterationen gemacht werden, entsprechend länger. (vgl. Rennhard, 2014)

2.1.2.1 SHA3

SHA3 ist 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 Geburtstagsparadoxon). (vgl. NIST, 2014)

2.1.2.2 PBKDF2

PBKDF2 steht für „Password Based Key Derivation Function 2“ und wurde entwickelt, um von einem gegebenen 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)

2.2 Dateiformate

2.2.1 INI

INI-Dateien werden zur Konfiguration von Applikationen verwendet. Es handelt sich dabei um eine Textdatei, welche verschiedene Wertepaare beinhaltet. Die Wertepaare bestehen aus einem Schlüssel und einem Wert. Schlüssel und Wert werden durch ein Gleichheitszeichen zusammengesetzt.

2.2.2 XML

Die Konfiguration von Initialisierungsparametern und der Import von Prüfungsdetails kann mithilfe von XML-Dateien umgesetzt werden. XML eignet sich gut zur hierarchischen Strukturierung von Daten. 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 Wurzelement.
- Zu jedem Start-Tag gehört ein End-Tag, das 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 können XML-Dateien eingelesen, geprüft und einer neuen Struktur übergeben werden.

2.2.3 ODT: Open Office Writer

Eine in OpenOffice Writer verfasste Prüfung wird als ODT Datei gespeichert. Dieses Format ist nichts anderes, als ein komprimierter Ordner, der die verfassten Inhalte im XML-Format enthält. Der Inhalt dieser XML-Datei basiert auf den Elementen von HTML.



	META-INF	08.12.2014 10:18	Dateiordner	
	styles.xml	06.11.2014 10:27	XML-Datei	20 KB
	settings.xml	06.11.2014 10:27	XML-Datei	3 KB
	meta.xml	06.11.2014 10:27	XML-Datei	1 KB
	content.xml	06.11.2014 10:27	XML-Datei	56 KB
	mimetype	06.11.2014 10:27	Datei	1 KB

Abbildung 5: Inhalt einer ODT-Datei

2.2.4 DOCX: Microsoft Office Word

Die Struktur von Word-Dokumenten ist ebenfalls im XML-Format, welches HTML ähnliche Tags enthält. 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.





	_rels	08.12.2014 10:29	Dateiordner	
	docProps	08.12.2014 10:29	Dateiordner	
	word	08.12.2014 10:29	Dateiordner	
	[Content_Types].xml		XML-Datei	5 KB

Abbildung 6: Inhalt einer DOCX-Datei

2.2.5 XSLT

XSLT ist eine Definition zur Umwandlung von XML-Dateien in andere Formen von XML-Dokumenten oder XML-ähnlichen-Formaten. Das Format von XSLT ist selbst in einer XML-Struktur definiert.

Zur Erstellung einer XML-Struktur mithilfe von XSLT wird ein zusätzliches Programm benötigt. In dieses wird dann sowohl das XML File als auch die definierte XSLT-Datei importiert. Anschliessend kann mit der XSLT Datei die XML Datei die auszugebende Struktur transformiert werden. Die Ausgabe wird an einen vordefinierten Ort gespeichert und kann für weitere Arbeiten verwendet werden.

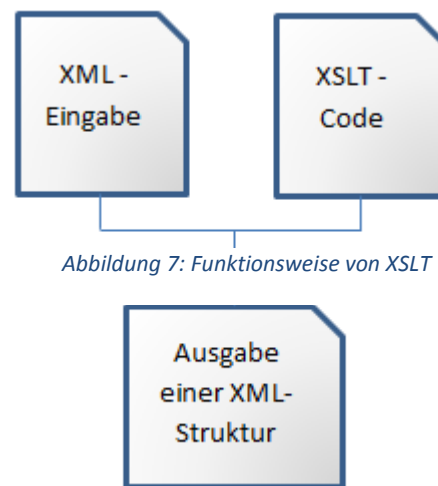


Abbildung 7: Funktionsweise von XSLT

2.2.6 PDF

Das Dateiformat für elektronische Dokumente dient dazu, Texte Bilder und Grafiken in einer Datei zusammenzufassen. Diese können unabhängig vom ursprünglichen Anwendungsprogramm, vom Betriebssystem oder von der Hardwareplattform weitergegeben werden, ohne dass, die Struktur oder Darstellung des Inhalts verändert wird. Des Weiteren kann eine PDF-Datei Steuerelemente, wie Textfelder und Buttons, aber auch JavaScript Code beinhalten. Das Dokument kann so gesichert werden, dass zum Beispiel beim Öffnen der Datei ein Passwort verlangt wird. (Adobe, 2006)

2.2.7 HTML

Die textbasierte Auszeichnungssprache HTML wird zur Strukturierung digitaler Dokumente verwendet. Ihre Hauptaufgabe ist die Darstellung von Inhalten im Internet, welche von Webbrowsern angezeigt werden können. Zum Inhalt gehören Texte, Links, Bilder und andere Elemente, welche auf Webseiten zum Einsatz kommen. Die Sprache wird dazu verwendet, einen Text mit den benötigten Elementen zu strukturieren, aber nicht zu gestalten oder Funktionalitäten auszuführen. Die visuelle Darstellung wird durch CSS realisiert, während die Interaktionen zwischen dem Benutzer und der Seite mithilfe von JavaScript umgesetzt werden kann. Die aktuelle Version ist seit dem 28. Oktober 2014 HTML5, welche von den meisten Browsern unterstützt wird. (vgl. Hickson, et al., 2014)

2.3 Digitale Manipulationsmöglichkeiten

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

2.3.1 Korrekte Antworten auslesen

Angenommen, zur Korrektur der Antworten sind bereits die korrekten Antworten in der Prüfung gespeichert, dann wäre es mit genügend Fachwissen möglich, diese auszulesen und somit die Prüfung vollständig korrekt zu beantworten.

2.3.2 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:

2.3.2.1 Manipulation der geräteinternen Uhr

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

$$(\text{Startzeit} + \text{Prüfungsdauer}) - \text{aktuelle Uhrzeit}$$

Dann wäre es möglich, nach dem Starten der Prüfung, die interne Uhr um eine gewisse Zeitspanne zurückzusetzen um mehr Zeit zum Lösen der Prüfung zu erhalten.

2.3.2.2 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.

2.3.3 Internet Zugriff

Wireless Module sind in praktisch allen Notebooks und Tablets eingebaut. Es wäre möglich, während der Prüfung im Internet nach Lösungen der Prüfungsfragen zu suchen, wenn zum Beispiel das eigene Mobiltelefon als Internet Gateway verwendet werden.

2.3.4 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:

2.3.4.1 Wireless LAN

Auch wenn der Prüfungsraum gegen jegliche Wirellessstrahlung 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.

2.3.4.2 Bluetooth Modul

Mittels Bluetooth sind Datenverbindungen zwischen zwei Endgeräten möglich. Somit könnten Kandidaten durch Einschalten von Bluetooth miteinander kommunizieren.

2.4 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.

2.4.1 Einschränkungen

2.4.1.1 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.

2.4.1.2 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.

2.4.1.3 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 unterschiedlichen Lichtverhältnissen.

3 Analyse

3.1 Anwendungsfälle

3.1.1 Übersicht

- AF01: Prüfung erstellen
- AF02 Prüfung absolvieren

3.1.2 Akteure

3.1.2.1 Dozent

Der Dozent hat die Berechtigung eine Prüfung zu erstellen und kennt die einzelnen Angaben zu den Studenten.

3.1.2.2 Student

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

3.1.2.3 System

Als System wird die Applikation selbst bezeichnet.

3.1.3 AF01: Prüfung erstellen

3.1.3.1 Kurzbeschreibung

Dem Dozenten 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 generiert.

3.1.3.2 Akteure

Dozent, System

3.1.3.3 Auslöser

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

3.1.3.4 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.

3.1.3.5 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.

3.1.3.6 Benutzte Anwendungsfälle

Keine

3.1.3.7 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

3.1.3.8 Spezielle funktionale Anforderungen

Die Parameter müssen klar definiert sein und falls ein Fehler auftritt, muss eine aussagekräftige Meldung erscheinen. Enthalten die Konsolenparameter einen Fehler muss die richtige Verwendung dieser Applikation als Hilfestellung angezeigt werden.

3.1.3.9 Spezielle technische Anforderungen

Keine

3.1.3.10 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 ablaufen.

3.1.3.11 Datenspeicherung

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

3.1.3.12 Priorität

Hohe Priorität

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

3.1.4 AF02 Prüfung absolvieren

3.1.4.1 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.

3.1.4.2 Akteure

Student, System

3.1.4.3 Auslöser

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

3.1.4.4 Vorbedingung

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

3.1.4.5 Ergebnisse und Nachbedingungen

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

3.1.4.6 Benutzte Anwendungsfälle

„AF01: Prüfung erstellen“

3.1.4.7 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, die Antworten exportiert und die Manipulation festgehalten
4.a	System	Zeit abgelaufen
4.b	System	Prüfung wird beendet und die Antworten exportiert

3.1.4.8 Spezielle funktionale Anforderungen

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

3.1.4.9 *Spezielle technische Anforderungen*

Der Anmeldevorgang der Prüfung darf nicht länger als drei Sekunden dauern. Bei längerer Ladezeit ohne entsprechende Rückmeldungen, leidet die Usability des Produktes (siehe NFA03).

3.1.4.10 *Häufigkeit*

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

3.1.4.11 *Datenspeicherung*

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

3.1.4.12 *Priorität*

Hohe Priorität

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

3.2 Nichtfunktionale Anforderungen

3.2.1 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.

3.2.2 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.

3.2.3 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.

4 Konzept

4.1 Systemüberblick

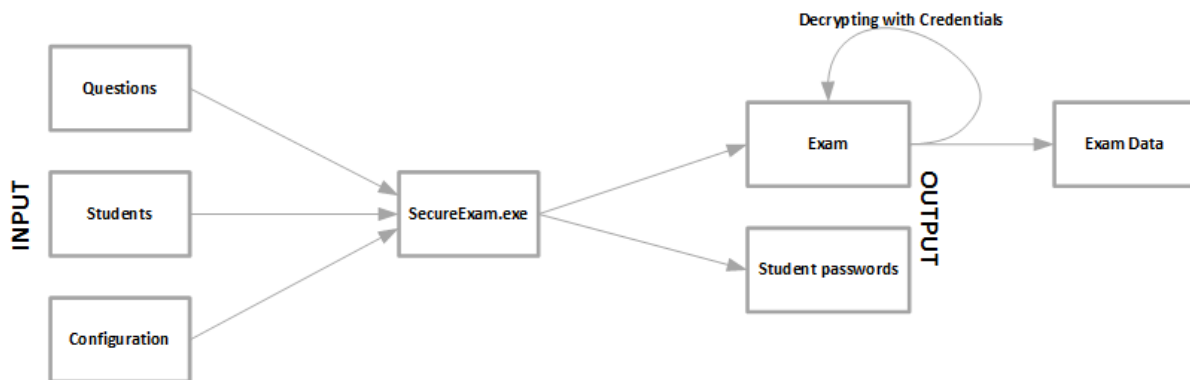


Abbildung 8: Systemüberblick

Das Diagramm bietet eine Übersicht über die Komponenten. SecureExam 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. Falls diese korrekt eingegeben wurden, entschlüsselt sich die Prüfung selber und die Fragen werden angezeigt. Beim Abschluss der Prüfung werden die Antworten exportiert.

4.1.1 Domänenmodell

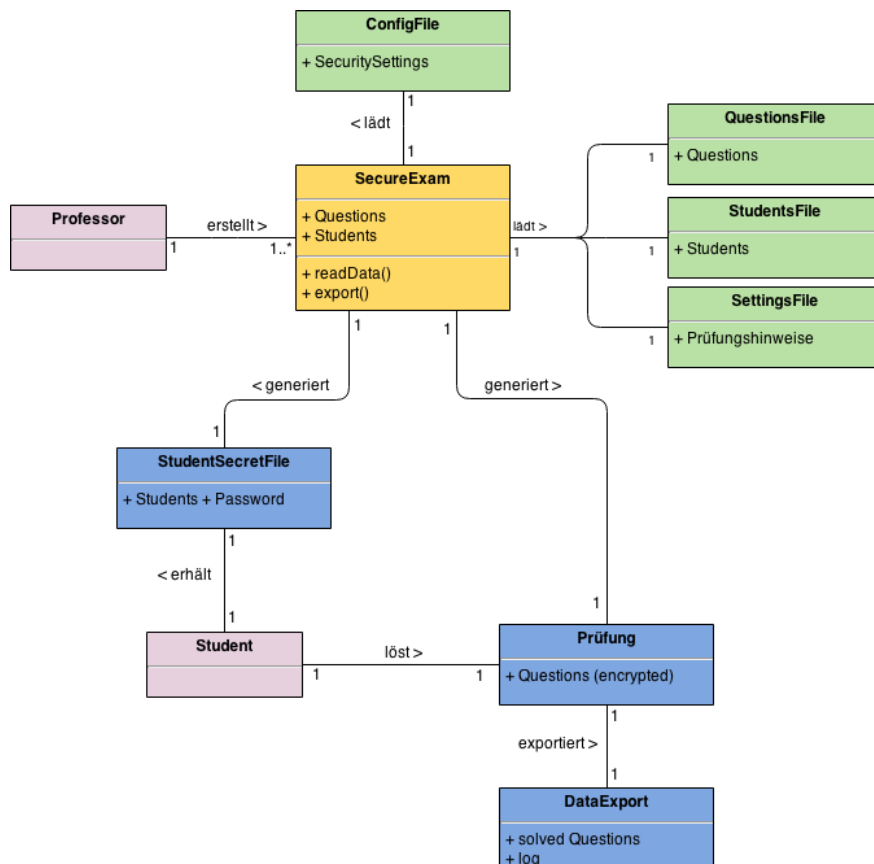


Abbildung 9: Domänenmodell SecureExam

4.2 C# Errorhandling

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.

```
try
{
    DateTime examDateTime = Convert.ToDateTime(examDateString);
    DateTime examStartTime = Convert.ToDateTime(examStartTimeString);
    DateTime examEndTime = Convert.ToDateTime(examEndTimeString);

    examStartTime = new DateTime(examDateTime.Year, examDateTime.Month, examDateTime.Day, examStartTime.Hour, examStartTime.Minute, 0);
    examEndTime = new DateTime(examDateTime.Year, examDateTime.Month, examDateTime.Day, examEndTime.Hour, examEndTime.Minute, 0);

    DataProvider.GetInstance().examDetails.examStartTime = examStartTime;
    DataProvider.GetInstance().examDetails.examEndTime = examEndTime;
}
catch (Exception e)
{
    throw new InvalidTimeException("Invalid data format");
}
```

Abbildung 10: Beispiel eines Exception rethrowings

4.3 Codedokumentation

Jede Funktion im Code wird mittels standardisierten Kommentaren dokumentiert. So kann die Dokumentation zusätzlich direkt in der IDE bei der Code-Vervollständigung angezeigt werden.

4.3.1 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.

4.3.2 JavaScript

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

4.4 Kryptographie

4.4.1 Verschlüsselung

4.4.1.1 Evaluationsmatrix

Kriterium	Gewicht	AES	RSA	AES gewichtet	RSA gewichtet
Sicherheit (bekannte Schwachstellen)	3	10	10	30	30
einfache Keyeingabe	5	10	1	50	5
einfache Mult-User Verschlüsselung	5	7	8	35	40
JavaScript Library verfügbar	4	10	10	40	40
C# Library verfügbar	4	10	10	40	40
Total ungewichtet		47	39		
Total gewichtet		195	155		

Gewicht:	1 (unwichtig) - 5 (sehr wichtig)
Punkte:	1 (sehr schlecht) - 10 (sehr gut)

Tabelle 1: Vergleich AES und RSA

4.4.1.2 *Entscheid*

Wie bei der Evaluationsmatrix ersichtlich, hat AES gegenüber RSA einen 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.

4.4.1.3 *Konzept*

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

4.4.1.3.1 *Verschlüsselung*

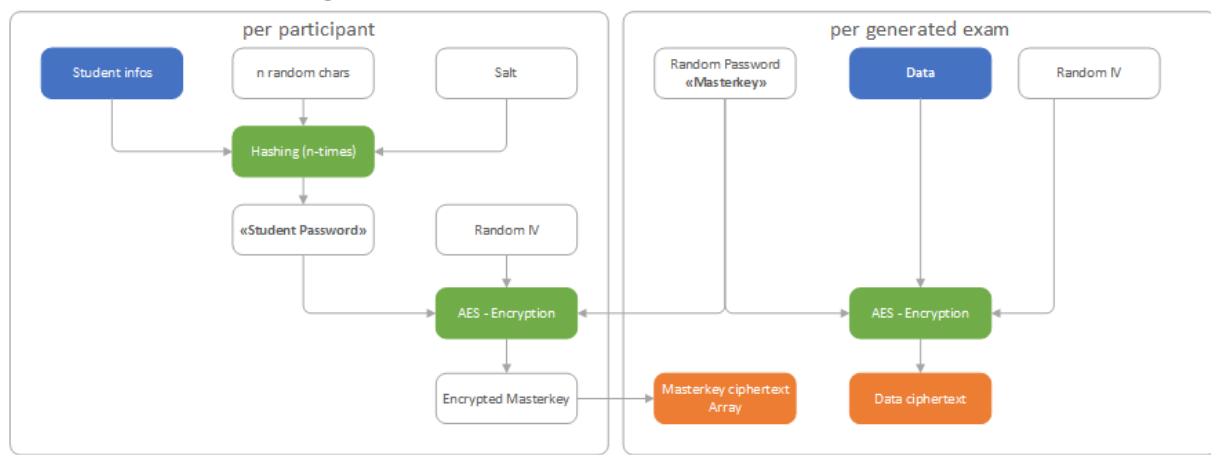


Abbildung 11: Multi-User AES Verschlüsselung

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

- Vorname
- Nachname
- Immatrikulationsnummer
- n zufällige Zeichen (Base64)

Diese werden anschliessend n-Mal mit gehasht (vgl. 4.4.2, Passwort hashing). Mit dem resultierenden Hash und einem zufälligen IV 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.

4.4.1.3.2 Entschlüsselung

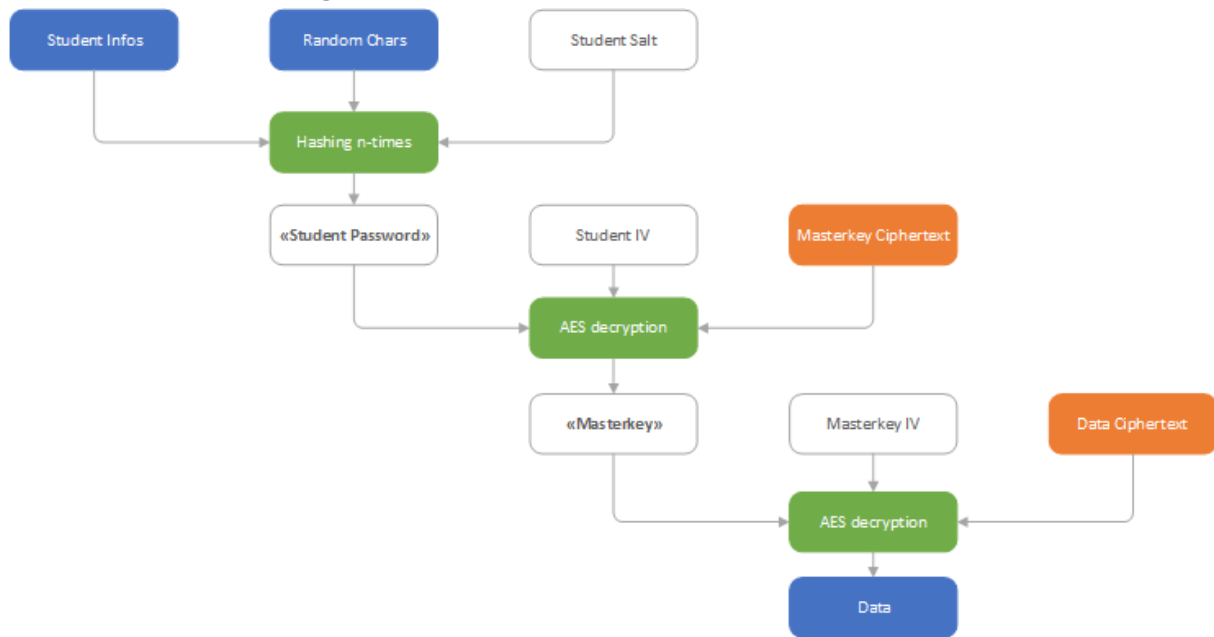


Abbildung 12: Multi-User AES Entschlüsselung

Jeder Student wird beim Starten der Prüfung seine Zugangsdaten eingeben müssen. 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 Prüfungsdaten entschlüsselt.

4.4.2 Passwort hashing

4.4.2.1 Evaluationsmatrix

Kriterium	Gewicht	SHA3	PBKDF2	SHA3 gewichtet	PBKDF2 gewichtet
Sicherheit (bekannte Schwachstellen)	5	10	10	50	50
Skalierbarkeit	5	9	6	45	30
JavaScript Library verfügbar	3	10	10	30	30
C# Library verfügbar	3	10	10	30	30
Total ungewichtet		39	36		
Total gewichtet		155	140		

Gewicht:	1 (unwichtig) - 5 (sehr wichtig)
Punkte:	1 (sehr schlecht) - 10 (sehr gut)

Tabelle 2: Vergleich SHA3 und PBKDF2

4.4.2.2 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.

4.4.2.3 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 definierbar.

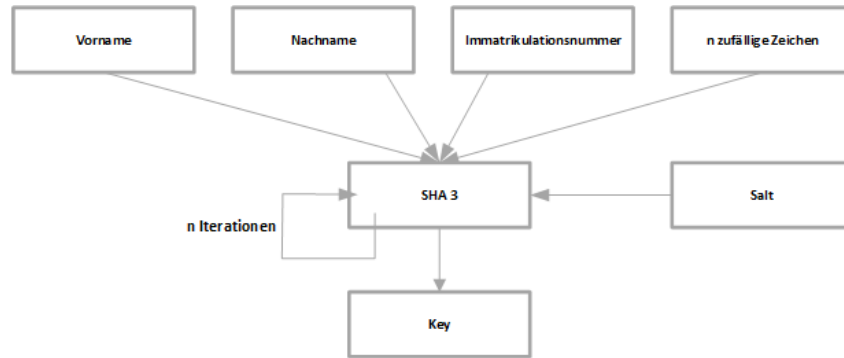


Abbildung 13: Hashing Ablauf

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.

4.5 Konfigurationsdateien

4.5.1 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. Deshalb wurde der Entscheid gefällt, XML einzusetzen.

4.5.2 Konzept

Es werden zwei verschiedene Konfigurationsdateien verwendet, um die Einstellungen bezüglich der Prüfung von den Einstellungen des Programmes abzugrenzen.

4.6 Import

4.6.1 Evaluationsmatrix

Kriterium	Gewicht	ODT	Word	ODT gewichtet	Word gewichtet
Brauchbares Format	5	8	4	40	20
Brauchbar beim Erstellen(Designen)	4	7	8	28	32
Kosten	1	10	1	10	1
Verbreitung	2	5	10	10	20
Total ungewichtet		30	23		
Total gewichtet		88	73		

Gewicht:	1 (unwichtig) - 5 (sehr wichtig)
Punkte:	1 (sehr schlecht) - 10 (sehr gut)

Tabelle 3: Vergleich ODT und Word

4.6.2 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 Checkboxes und Textfelder in Word so verschachtelt, dass das Parsen eines solchen Files aufwändiger zu realisieren wäre. Der Aufbau eines Open Office Dokuments eignet sich besser für unser Projekt, daher wurde entschieden einen Parser für ODT einzubauen.

4.6.3 Konzept

Es werden zwei verschiedene Methoden Angeboten die Prüfungsfragen zu importieren. Eine Variante ist, ein XML-File zu importieren. Die XML-Struktur ist vordefiniert, um das Parsen zu vereinfachen.

Die zweite Variante wird sein, ein ODT-Dokument anzugeben, welches mithilfe von XSLT zu einer verwertbaren XML-Struktur transformiert wird. Anschliessend wird dieses mit dem bereits erwähnten XML-Parsers eingelesen.

Zusätzlich müssen die Angaben zu den Prüfungsteilnehmern importiert werden. Diese werden in einer eigenen XML Datei erfasst und mit einem speziellen Parser eingelesen.

4.7 Export

4.7.1 Evaluationsmatrix

Kriterium	Gewicht	HTML	PDF	HTML gewichtet	PDF gewichtet
Plattformunabhängigkeit	5	10	10	50	50
Einfache Handhabung	2	10	10	20	20
Kosten	1	10	10	10	10
Darstellungsmöglichkeiten	1	9	8	9	8
Erfahrungen im Team	4	9	1	36	4
Total ungewichtet		48	39		
Total gewichtet		125	92		

Gewicht:	1 (unwichtig) - 5 (sehr wichtig)
Punkte:	1 (sehr schlecht) - 10 (sehr gut)

Tabelle 4: Vergleich HTML und PDF

4.7.2 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 besseren Kenntnisse des Teams HTML als Exportformat gewählt.

4.7.3 Konzept

Zum Exportieren von Daten in eine HTML-Struktur wird ein HTML Skeleton eingesetzt. Dieses Grundgerüst beinhaltet bereits viele vordefinierte Elemente sowie CSS und JavaScript Funktionen sowie einige Platzhalter. Diese werden beim Exportieren durch die entsprechenden Datensätze ersetzt.

Die Login Daten müssen separat gespeichert werden, da diese zum Entschlüsseln der Prüfung gebraucht werden. Deshalb werden diese separat in eine XML Datei exportiert.

4.8 Digitale Manipulationsmöglichkeiten

4.8.1 Zeit

4.8.1.1 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 die aktuelle Uhrzeit gegen den Verlauf geprüft.

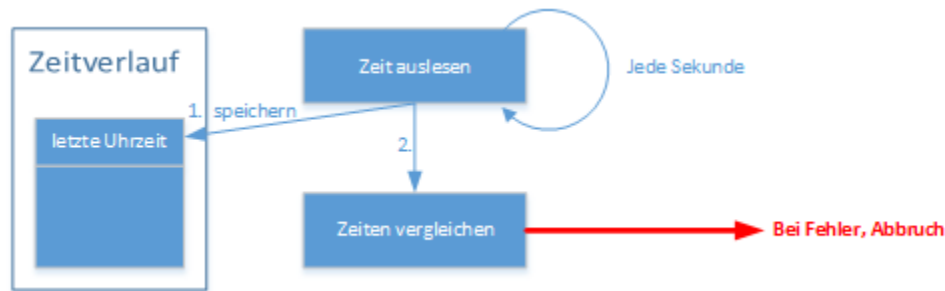


Abbildung 14: Manipulation der internen Uhr detektieren

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

4.8.1.2 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 wurde ein neues Konzept erarbeitet:

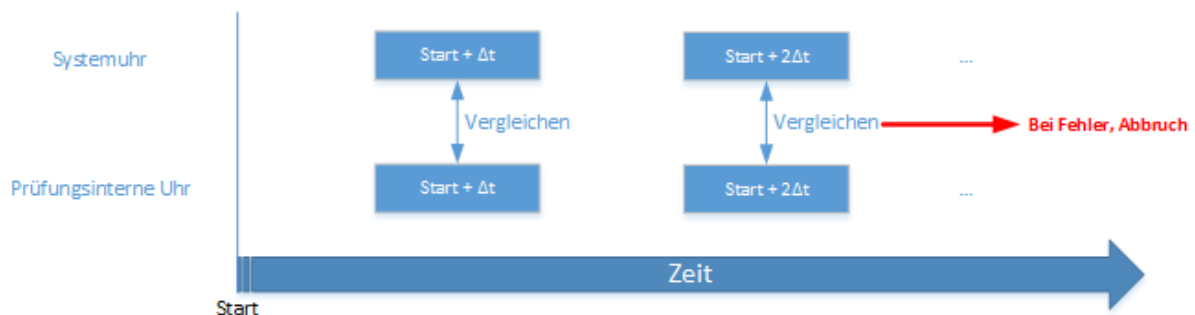


Abbildung 15: Vergleich der internen Uhr mit der Systemuhr

Nachdem die Prüfung gestartet wird, wird eine prüfungsinterne, in JavaScript realisierte Uhr gestartet. Diese läuft parallel zu der Systemuhr. Durch Vergleichen der beiden Uhrzeiten können Abweichungen sehr einfach festgestellt werden.

4.8.2 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.

4.8.3 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.

4.9 UI-Design

4.9.1 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 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.

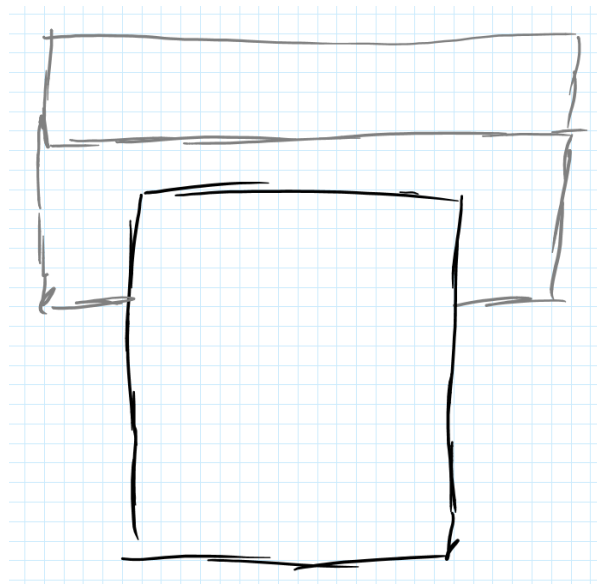


Abbildung 16: UI Design Idee

In der Handskizze sind zwei Blätter zu erkennen. Das graue Blatt im Querformat wird oben an der Prüfung sein und wichtige 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.

4.9.2 Login

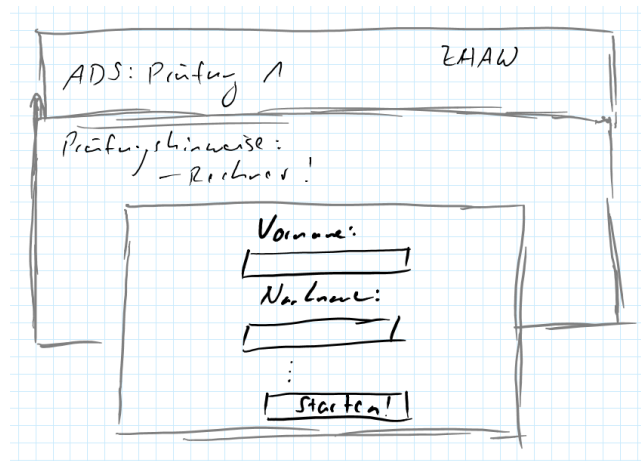


Abbildung 17: Login-Skizze

Nach dem Starten der Prüfung muss sich der Student authentifizieren. Es gibt vier Eingabefelder, die der Prüfungsteilnehmer ausfüllen muss:

- Vorname
- Name
- Immatrikulationsnummer
- Passwort

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

4.9.3 Prüfung

4.9.3.1 Page-Mode

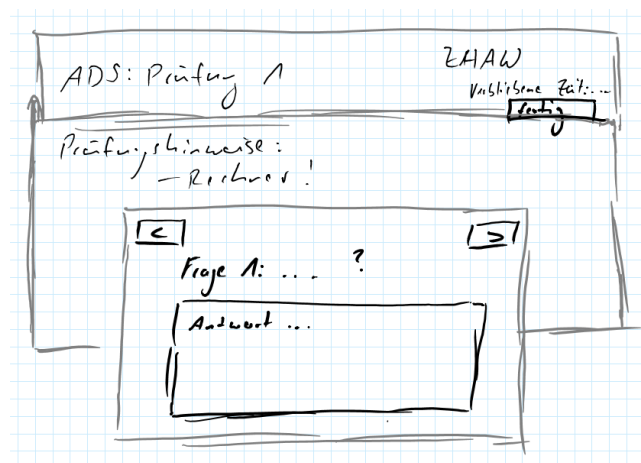
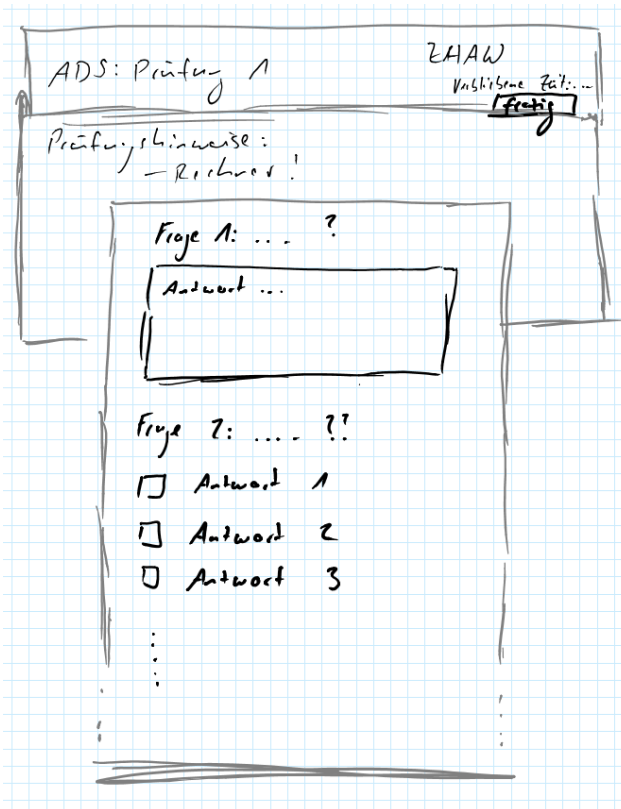


Abbildung 18: Skizze Paging Mode

Der Page Mode ist dafür gedacht, wenn die Prüfung auf eBook Readern mit geringer Bildwiederhol- frequenz absolviert wird. Zwischen den Fragen wird, ähnlich wie bei einem Buch, geblättert. Somit wird die Problematik des Scrollens umgangen.

4.9.3.2 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 die Kopfzeile konstant mit.

Abbildung 19: Skizze Scroll Mode

4.10 JavaScript

4.10.1 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.

4.10.2 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 und für ältere Modelle wird eine data:URI erstellt.

4.10.3 Code Obfuscation

JavaScript Code wird nicht kompiliert und ist daher sehr anfällig gegen Manipulationen. Ein einfacher Text-Editor genügt, um massive Eingriffe in die Programmlogik vorzunehmen. Selbst auf eBook Rea-

der 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

```
105 ▼ this.checkIfLoggerAvailable = function (errorLevel) {
106 ▼   for (var i = errorLevel; i < that.loggers.length; i++) {
107 ▼     if (that.loggers[i].length >= 1) {
108       return true;
109     }
110   }
111   return false;
112 }
113
114 ▼ return {
115   log: function (msg, sender, errorLevel) {
116     that.logToAll(msg, sender, errorLevel);
117   },
118   addLogger: function (logger, errorLevel) {
119     that.loggers[errorLevel].push(logger);
120   },
121   removeLogger: function (logger) {
122     for (var i = 0; i < that.loggers.length; i++) {
123       if (that.loggers[i] === logger) {
124         that.loggers.splice(i, 1);
125       }
126     }
127   },
128   ErrorLevel: that.ErrorLevel
129 }
```

Mit Obfuscation

```
6C\x65\x60\x65\x6E\x74\x73\x42\x79\x43\x6C\x61\x73\x73\x4E\x61\x6
64\x64\x65\x64\x20\x6C\x69\x73\x74\x65\x6E\x65\x72\x20\x74\x6F\x2
8\x50\x49\x52\x45\x44", "\x72\x65\x60\x6F\x76\x65\x64\x20\x6C\x69\
x20\x66\x72\x6F\x60\x20\x41\x55\x54\x4F\x53\x41\x56\x45", "\x72\x6
3\x65\x74\x49\x6E\x74\x65\x72\x6E\x61\x6C\x54\x69\x60\x65\x40\x61
x65\x60\x6F\x76\x65\x49\x74\x65\x60"; _0x24db[0]; Date[_0x24db[2]]
(this[_0x24db[7]]() < 10)? _0x24db[5]+this[_0x24db[7]](): this[_0x24d
this[_0x24db[3]]+_0x24db[10]+this[_0x24db[6]]+_0x24db[10]+this[_0
{}]; SecureExam[_0x24db[13]]={}; SecureExam[_0x24db[13]][_0x24db[14]
{}]; SecureExam[_0x24db[19]]={}; SecureExam[_0x24db[19]][_0x24db[20]
[_0x24db[23]]=100000; SecureExam[_0x24db[16]][_0x24db[24]]=_0x24db
[_0x24db[30]]=_0x24db[31]; SecureExam[_0x24db[16]][_0x24db[18]][_0
[_0x24db[36]]=_0x24db[37]; SecureExam[_0x24db[15]][_0x24db[38]]=0;
[_0x24db[42]]=4; SecureExam[_0x24db[15]][_0x24db[43]]=5; SecureExam
{info:2,warning:1,error:0}; this[_0x24db[47]]=[]; this[_0x24db[47]]
[]; this[_0x24db[51]]=function (_0xbb02x3, _0xbb02x4, _0xbb02x5){if(
()+_0x24db[57]; switch(_0xbb02x5){case this[_0x24db[46]][_0x24db[4
[_0x24db[50]]:_0xbb02x6+=_0x24db[60]; break ;;} ; _0xbb02x6+=_0x24d
_0xbb02x8=0; _0xbb02x8<_0xbb02x2[_0x24db[47]][_0xbb02x7][_0x24db[6
_0xbb02x8=_0xbb02x5; _0xbb02x8<_0xbb02x2[_0x24db[47]][_0x24db[62]]
(_0xbb02x3, _0xbb02x4, _0xbb02x5){_0xbb02x2[_0x24db[51]](_0xbb02x3,
(_0xbb02x9){for(var _0xbb02x8=0; _0xbb02x8<_0xbb02x2[_0x24db[47]][
,ErrorLevel:_0xbb02x2[_0x24db[46]]];}); SecureExam[_0x24db[19]][_
(_0xbb02xa); this[_0x24db[69]]=document[_0x24db[68]](_0xbb02xb); th
_0xbb02x2=this; _0xbb02x2[_0x24db[73]]=new
```

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.

5 Umsetzung

5.1 Kryptographie

5.1.1 Verschlüsselung

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

```
5 using System.Linq;
6 using System.Security.Cryptography;
7 using System.Diagnostics;
```

Abbildung 20: Kryptographie Import in C#

Alle symmetrischen 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.

```
using (Aes aesAlg = Aes.Create())
{
    aesAlg.Key = Key;
    aesAlg.IV = IV;

    ICryptoTransform encryptor = aesAlg.CreateEncryptor(aesAlg.Key, aesAlg.IV);

    using (MemoryStream msEncrypt = new MemoryStream())
    {
        using (CryptoStream csEncrypt = new CryptoStream(msEncrypt, encryptor, CryptoStreamMode.Write))
        {
            using (StreamWriter swEncrypt = new StreamWriter(csEncrypt))
            {
                swEncrypt.Write(data);
            }
            return msEncrypt.ToArray();
        }
    }
}
```

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 einfachen Ablesbarkeit der realen Werte.

5.1.2 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.

```
767 // decrypt masterkey
768 ▼ var masterKeyCipherParams = CryptoJS.lib.CipherParams.create({
769     ciphertext: masterkeyCypher,
770     key: key,
771     iv: masterkeyIV,
772     algorithm: CryptoJS.algo.AES,
773     mode: CryptoJS.mode.CBC,
774     padding: CryptoJS.pad.PKCS7,
775     blockSize: 4,
776     formatter: CryptoJS.format.OpenSSL
777 });
778
779 ▼ var decMasterKey = CryptoJS.AES.decrypt(masterKeyCipherParams, key, {
780     iv: masterkeyIV
781 });
782 var masterKeyString = decMasterKey.toString(CryptoJS.enc.Utf8);
```

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 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 IV angegeben werden.

5.1.3 Passwort hashing

5.1.3.1 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.

```
50 using(SHA256 mySHA256 = SHA256Managed.Create())
51 {
52     String ivB64 = Convert.ToBase64String(iv);
53
54     byte[] hash = mySHA256.ComputeHash(Encoding.UTF8.GetBytes(data + ivB64));
55     for( int i = 0; i < iterations -1; i++ )
56     {
57         hash = mySHA256.ComputeHash(hash);
58     }
59
60     return hash;
61 }
```

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 Konfiguration eingestellt.

5.1.3.2 JavaScript

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

```
760     var userSecretSalted = that.User.toString() + saltB64;  
761     var key = CryptoJS.SHA256(userSecretSalted);  
762     for (var i = 0; i < SecureExam.Const.Cryptography.SHA256ITERATIONS - 1; i++) {  
763         key = CryptoJS.SHA256(key);  
764     }
```

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.

5.2 Konfigurationsdateien

Zum Einlesen der Konfigurationsdateien 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.

5.2.1 Prüfungskonfigurationsdatei

Die Datei zur Angabe von Konfigurationen bezüglich der Prüfung wird über die Konsolenparameter angegeben.

```
<settings>  
  <subject>ADS</subject>  
  <examTitle>Algorithmen und Datenstrukturen</examTitle>  
  <examNotes>Keine Hilfsmittel erlaubt</examNotes>  
  <examDate>1.12.2014</examDate>  
  <startTime>14:00</startTime>  
  <endTime>16:00</endTime>  
  <duration>45</duration>  
  <internetAllowed>false</internetAllowed>  
  <tabChangeAllowed>true</tabChangeAllowed>  
  <historyTimeMaxVariance>5000</historyTimeMaxVariance>  
  <internalTimeMaxVariance>5000</internalTimeMaxVariance>  
  <confirmAutosaveRestore>true</confirmAutosaveRestore>  
  <ebookreaderExport>true</ebookreaderExport>  
  <viewMode>scroll</viewMode>  
</settings>
```

Abbildung 25: Beispiel einer Prüfungskonfigurationsdatei

Es existieren 14 verschiedene Parameter, welche in der Bedienungsanleitung beschrieben werden. Wird ein obligatorischer Parameter nicht gesetzt, erscheint beim Erstellen der Prüfung eine Fehlermeldung.

5.2.1.1 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.

```
<examNotes>
  <h3>Prüfungshinweise:</h3>
  <ul>
    <li>Es sind mehrere Antworten möglich</li>
    <li>Rechner erlaubt!</li>
  </ul>
</examNotes>
```

Prüfungshinweise:

- Es sind mehrere Antworten möglich
- Rechner erlaubt!

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

5.2.2 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.

```
<data>
  <NumberOfRandomCharsInStudentSecret>10</NumberOfRandomCharsInStudentSecret>
  <AESSettings>
    <keyLength>256</keyLength>
    <ivLength>128</ivLength>
  </AESSettings>
  <SHA256Settings>
    <iterations>100000</iterations>
    <saltLength>256</saltLength>
  </SHA256Settings>
</data>
```

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 27 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.

5.3 Import

Über Konsolenparameter müssen die Import-Dateien angegeben werden. Der Dateityp dieser 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. Die Applikation wurde so aufgebaut, dass weitere Importmöglichkeiten jederzeit hinzugefügt werden können.

5.3.1 XML

Der Import über eine XML-Datei erfolgt über einen in C# entwickelten Parser, der das XML Framework von .NET verwendet. Die Tags 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.

```
<questions>
  <question>
    <legend>Frage 1. Welche der folgenden Operationen kommen in einem Stack vor?</legend>
    <input type="checkbox" value="enter" />
    <input type="checkbox" value="top" />
    <input type="checkbox" value="empty" />
    <input type="checkbox" value="front" />
    <input type="checkbox" value="push" />
  </question>
</questions>
```

Abbildung 28: XML-Auszug mit einer Frage

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

5.3.2 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.

```
<xsl:template match="form:checkbox">
  <xsl:element name="input">
    <xsl:attribute name="type">checkbox</xsl:attribute>
    <xsl:attribute name="value"><xsl:value-of select="@form:label" /></xsl:attribute>
  </xsl:element>
</xsl:template>
```

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. 5.3.1 XML) übergeben werden kann. Nach diesem letzten Arbeitsvorgang sind die Daten im Programm angekommen und der Import ist dadurch abgeschlossen.

5.3.3 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.

```
<data>
  <professor>
    <name>Rege</name>
    <vorname>Karl</vorname>
  </professor>
  <students>
    <student>
      <name>Jampen</name>
      <vorname>Daniel</vorname>
      <number>S12198320</number>
    </student>
    <student>
      <name>Lukes</name>
      <vorname>Simon</vorname>
      <number>S12198347</number>
    </student>
  </students>
</data>
```

Abbildung 30: Beispiel von Einträgen im Studenten Import File

5.4 Export

5.4.1 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 Skeleton-Datei. Während CSS und JavaScript bereits im Skeleton-File integriert sind, müssen die Prüfungsdetails noch generiert werden.

```
<div id="autoSaveInfo"></div>
<div class="headerTitle">
  $SUBJECT$: $EXAMTITLE$
  <p id="studentDetails" class="headerStudentInfo"></p>
</div>
```

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.

```
html = html.Replace("$SUBJECT$", DataProvider.GetInstance().examDetails.subject);
html = html.Replace("$EXAMTITLE$", DataProvider.GetInstance().examDetails.examTitle);
```

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

5.4.2 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.

```
<data>
  <professor>
    <name>Rege</name>
    <vorname>Karl</vorname>
  </professor>
  <students>
    <student>
      <name>Lukes</name>
      <vorname>Simon</vorname>
      <number>S12198347</number>
    </student>
  </students>
</data>
```

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.

5.5 Digitale Manipulationsmöglichkeiten

5.5.1 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 CPU-Lag-Zeiten Analysiert werden:

Iteration	Fenster aktiv	Fenster aktiv + hohe CPU last	Fenster inaktiv	Fenster inaktiv + hohe CPU last
1	16 ms	1 ms	246 ms	661 ms
2	1 ms	2 ms	996 ms	38 ms
3	17 ms	3 ms	1005 ms	961 ms
4	1 ms	2 ms	990 ms	1 ms
5	2 ms	1 ms	7 ms	7 ms
6	14 ms	1 ms	1009 ms	1000 ms
7	8 ms	3 ms	988 ms	994 ms
8	1 ms	2 ms	1014 ms	8 ms
9	8 ms	1 ms	996 ms	991 ms
10	3 ms	2 ms	985 ms	16 ms
11	12 ms	2 ms	972 ms	978 ms
12	7 ms	3 ms	1027 ms	26 ms
13	6 ms	3 ms	978 ms	989 ms
14	1 ms	1 ms	280 ms	980 ms
15	12 ms	1 ms	890 ms	34 ms
Durchschnitt:	7 ms	2 ms	826 ms	512 ms

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 werden folgende Schlüsse gezogen:

- Ist das Browserfenster mit dem JavaScript Code aktiv, treten Aufrufverzögerungen < 20ms auf.
- Ist das Browserfenster inaktiv, d.h. der Benutzer surft auf einem anderen Fenster, beträgt die Verzögerung in etwa eine Sekunde.
- Eine hohe CPU Last bewirkt bei aktivem und inaktivem Fenster eine genauere Ausführung der setInterval Methode. Wobei dieser Effekt bei inaktivem Fenster nur etwa jede zweite Sekunde auftritt.
- 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:

- 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.
- Browserfenster-Fokuswechsel erlauben und die Abweichung auf >2s stellen. Dies hebt 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:

```
22 <tabChangeAllowed>true</tabChangeAllowed>
23 <historyTimeMaxVariance>5000</historyTimeMaxVariance>
24 <internalTimeMaxVariance>5000</internalTimeMaxVariance>
```

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

In der nachfolgenden Abbildung erkennt man die Zeitvergleichsfunktion, die Akzeptanz eines gewissen CPU Lags (variance Variable) sowie die Codestelle zur Kompensierung des CPU Lags. Da die Systemzeit vorgängig geprüft wird, wird davon ausgegangen, dass die Systemzeit valid ist. Nun wird die interne Zeit um die Varianz zur Systemzeit nachgestellt und läuft wieder genau.

```
381 // verify InternalTime
382 if (that.dateCompare(internalTime, systemTime)) {
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414 this.dateCompare = function (actualTime, timeToVerify, maxVariance) {
415     var maxVariance = (typeof maxVariance !== "undefined") ? maxVariance : that.INTERNALCLOCKMAXVARIANCE;
416     timeToVerify = new Date(timeToVerify);
417     actualTime = new Date(actualTime);
418
419     var variance = timeToVerify.getTime() - actualTime.getTime();
420     if (variance <= maxVariance && variance >= -maxVariance) {
421         return true;
422     }
423     return false;
424 }
425
426
427
428
429
430
431
432
433 // compensate cpu lag in internal Time (systemTime is more accurate)
434 var diff = systemTime.getTime() - internalTime.getTime();
435 if (diff > 0) {
436     that.internalClockStartTime.setTime(that.internalClockStartTime.getTime() + diff);
437     SecureExam.Logger.log("compensing cpu lag: " + diff + "ms", "secureDate", SecureExam.Logger.ErrorLevel.warning);
438 }
```

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

5.5.2 Internet Zugriff

Mittels JavaScript kann sehr einfach ein neues Bild erzeugt und 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. Das bedeutet, dass die Internetverbindung nicht aktiv ist.

```

226   this.check = function () {
227       SecureExam.Logger.log("checking internet", "InternetAccessCheck", SecureExam.Logger.ErrorLevel.info);
228       var img = new Image();
229       img.onload = that.onLoad;
230       img.onerror = that.onError;
231       img.src = that.imgURL + '?t=' + new Date().getTime();
232   }
  
```

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.

5.6 UI-Design

5.6.1 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.

5.6.2 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!

5.6.3 Page-Mode

Der Page-Mode ist für eBook Reader gedacht und zeigt jeweils nur eine Frage pro Seite an.



Abbildung 38: Prüfung im Page-Mode

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

5.6.4 Scroll-Mode

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

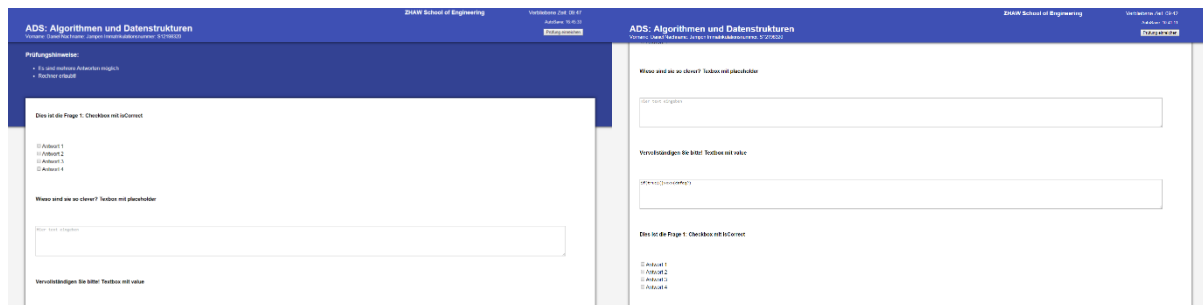


Abbildung 39: Prüfung im Scroll-Mode blendet die Hinweise aus

Wie auf der Grafik ersichtlich, verschwinden die Prüfungshinweise beim Scrollen nach unten. Dies wurde so gewählt, 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.

6 Testing

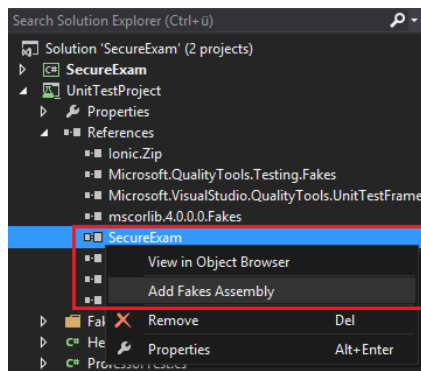
6.1 Konzept

Zum Überprüfen der Funktionalität von C# Code, eignen sich Unit-Tests besonders gut. Microsoft bietet bereits eine Testing-Suite, daher macht es Sinn diese zu verwenden anstatt zusätzliche Libraries einzubinden. Wichtig zu beachten ist, dass die Tests nur mit Visual Studio 2012/2013 Ultimate funktionieren, da bei den anderen Versionen die Funktionalität nicht enthalten ist.

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

6.2 Realisierung

6.2.1 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 ist dieser SecureExam.Fakes. Nun kann mittels Lambda Ausdrücken die Funktionalität einer Klasse bzw. Funktion simuliert werden. Die Klassen sind immer zusammengesetzt aus „Shim“ und dem Klassennamen.

```
SecureExam.Fakes.ShimFacade.AllInstances.exportOutputTypeStringStudentSecretsFileFormat = (a, b, c, d) => {
};
SecureExam.Fakes.ShimFacade.AllInstances.readDataQuestionFormularTypeStringStudentFileTypeStringString = (a, b, c, d, e, f) => {
    throw new NotImplementedException("");
};
```

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 direkt eine `NotImplementedException`.

So kann sehr einfach jegliches Verhalten von Klassen simuliert werden.

6.2.2 Konsolenausgabe abfangen

Oftmals ist es bei Tests für die Konsolenklasse nötig, die Konsolenausgabe abzufangen und zu überprüfen. Dies wird in C# so gelöst, dass man die Out Eigenschaft der Konsole überschreibt, um den Output auf einen eigenen StreamWriter umzuleiten. Dieser kann dann sehr einfach mit `.ToString()` ausgelesen werden.

```
288         using (StreamWriter sw = new StreamWriter())
289         {
290             Console.SetOut(sw);
291
292             // TestLogik
293
294             sw.ToString();
295
296             // Assert...
297
298             // fake
299         }
```

Abbildung 42: Konsolenausgabe umleiten in C#

6.3 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 decken nachfolgende Test-Resultate nur rund 50% des Quelltextes ab.

6.3.1 Unit Tests

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

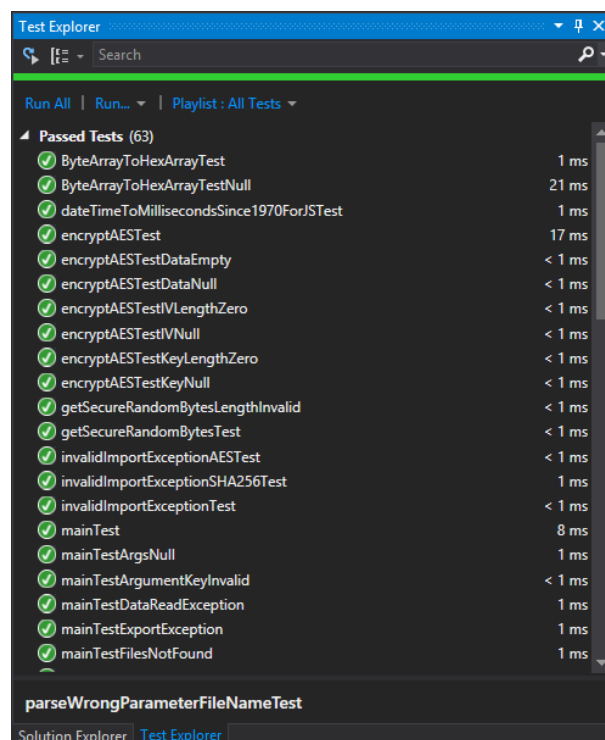
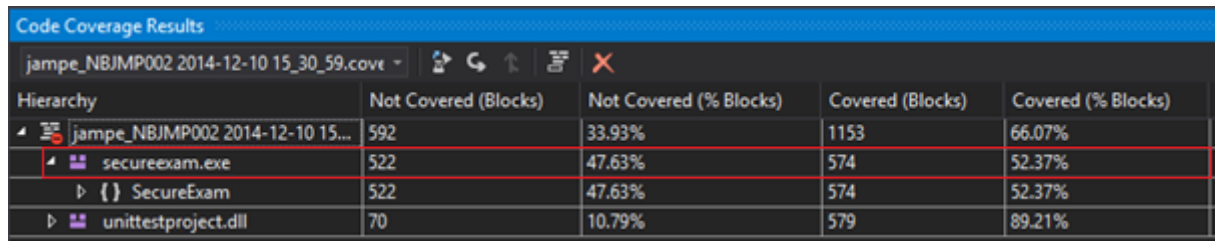


Abbildung 43: TestExplorer mit den Unit Tests

6.3.2 Code Coverage Analyzer



Hierarchy	Not Covered (Blocks)	Not Covered (% Blocks)	Covered (Blocks)	Covered (% Blocks)
jampe_NBJMP002 2014-12-10 15_30_59.cove	592	33.93%	1153	66.07%
secureexam.exe	522	47.63%	574	52.37%
SecureExam	522	47.63%	574	52.37%
unittestproject.dll	70	10.79%	579	89.21%

Abbildung 44: Auszug der Code Coverage Analyzer Ausgabe

Wie als Ziel festgelegt, wurde beim Testen 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.

7 Resultate

Als Resultate werden Bildschirmaufnahmen von der generierten Prüfung auf verschiedenen Geräten präsentiert. Es wurden drei Arten der Konfiguration, welche der Anwendung am ehesten entsprechen, sowohl für Tablets als auch für den herkömmlichen Desktop-PC zur Darstellung gewählt.

7.1 Prüfung auf PC mit Scroll-Funktion

Wird die Prüfung auf einem PC oder Notebook durchgeführt, so empfiehlt es sich die Scroll-Methode(siehe 5.6.4 Scroll-Mode) zu verwenden, da diese Geräte gut dafür geeignet sind, Bildschirmhalte zu verschieben.



Verbliebene Zeit: 18:00

ADS: Algorithmen und Datenstrukturen

Vorname: Simon Nachname: Lukes Immatrikulationsnummer: S12198347 Prüfung einreichen

Prüfungshinweise:

- Es sind mehrere Antworten möglich
- Rechner erlaubt!

Frage 1. Welche der folgenden Operationen kommen in einem Stack vor?

- ☐ enter
- ☐ top
- ☐ empty
- ☐ front
- ☐ push

Abbildung 45: Prüfung auf PC mit Scroll-Funktion

7.2 Prüfung auf portablen Geräten

7.2.1 Prüfung im Paging-Mode auf Tablet mit eInk Bildschirm

Unter der Verwendung eines eInk Bildschirms herrschen fast identische Voraussetzungen wie bei einem Tablet mit LCD Bildschirm, bis auf die Tatsache, dass wegen der unterschiedlichen Bildschirm-Technologie die Scrollfunktion nicht geeignet ist. Deshalb wurde der Paging-Mode (siehe 5.6.3 Page-Mode) entwickelt.

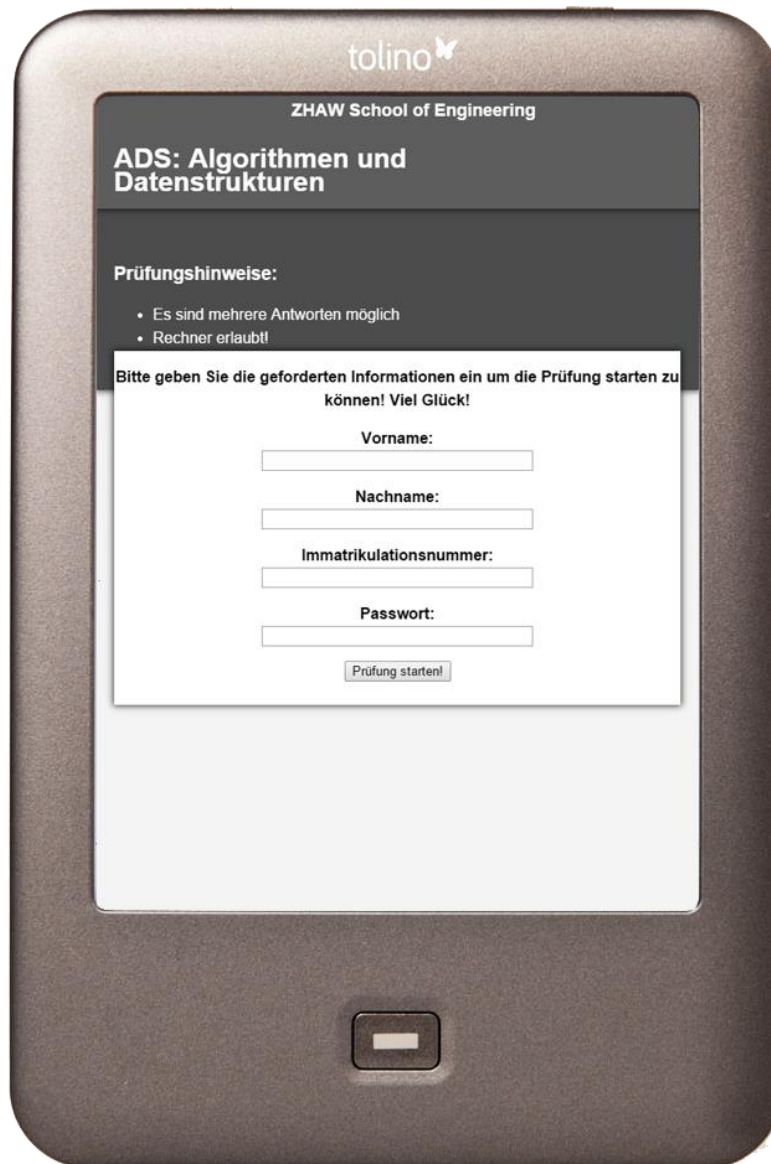


Abbildung 46: Prüfung im Paging-Mode auf Tablet mit eInk Bildschirm

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

Die Darstellung auf einem Tablet kann insofern von der Anzeige auf einem Desktop-PC unterschiedlich sein, dass die Auflösung auf dem Tablet möglicherweise kleiner ist und wegen des Platzmangels bei eingeschalteter Tastatur, meist im Hochformat verwendet wird. Dank dem Responsive Design von SecureExam, hindert weder das Format noch die Auflösung die Durchführung der Prüfung.

ZHAW School of Engineering Verbliebene Zeit: 19:00

ADS: Algorithmen und Datenstrukturen

Vorname: Simon Nachname: Lukes
Immatrikulationsnummer: S12198347

Prüfung einreichen

Prüfungshinweise:

- Es sind mehrere Antworten möglich
- Rechner erlaubt!

Zurück **Weiter**

Frage 1. Welche der folgenden Operationen kommen in einem Stack vor?

- ☐ enter
- ☐ top
- ☐ empty
- ☐ front
- ☐ push

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

8 Diskussion und Ausblick

8.1 Erreichte Ergebnisse

Die Applikation mit allen Funktionen, Import- und Exportmöglichkeiten wurden gemäss Aufgabenstellung oder durch Absprache 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 entwickeln wodurch 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, wenn auch in einem anderen Umfeld. Die Prüfung muss, anstatt auf Papier ausgedruckt, auf die eBook Reader kopiert und nach der Prüfung wieder davon extrahiert werden.

8.2 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 und somit nur diese gewählt werden kann. Diese Funktionalitäten können dank dem modularen Aufbau von SecureExam sehr einfach erweitert werden. Folgende Funktionalitäten könnten zum Beispiel 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 korrigiert und auswertet. Multiple-Choice Aufgaben eignen sich dafür am besten. Sobald Textbasierte Antworten möglich sind, wird die automatisierte Korrektur allerdings sehr schwierig umzusetzen.

Dieses Projekt trägt im Allgemeinen dazu bei, die Digitalisierung von Prüfungen voranzutreiben. Ein Hauptziel von SecureExam ist es, dass sowohl betrügen erschwert und auch Infrastrukturprobleme minimiert werden. Wir hoffen mit unseren Ansätzen und Ideen die Umsetzung von zukünftigen elektronischen Prüfungen erleichtert zu haben.

9 Verzeichnisse

9.1 Literaturverzeichnis

Adobe, 2006. *Adobe*. [Online]

Available at:

http://www.images.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/pdf_reference_1-7.pdf

[Zugriff am 13 Dezember 2014].

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].

Hickson, I. et al., 2014. *W3C HTML5*. [Online]

Available at: <http://www.w3.org/TR/2014/REC-html5-20141028/>

[Zugriff am 13 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.

9.2 Glossar

Begriff	Erklärung
Ciphertext	Ist die Ausgabe von einem Verschlüsselungsalgorithmus und ist für den Betrachter ohne Entschlüsselung nicht lesbar.
Hash	Das Resultat einer Hashberechnung hat die Länge von 2^x Bit, wobei meist grösser als 128 Bit.
Hash-chaining	Die mehrfache direkt aufeinanderfolgende Ausführung eines Hashalgorithmus.
Geburtstagsparadoxon	Bezeichnet das Paradoxon, das sich die Sicherheit auf Kollisionsangriffe halbiert wenn der kollidierende Hashwert nicht relevant ist.
ODT	Ist das Format, welches von Open Office Writer generiert wird.
IV	Initialisierungsvektor eines Verschlüsselungsalgorithmus. Wird ein gegebener Input ohne IV verschlüsselt, ist der Output immer gleich. Um eine Vorberechnung aller Outputs zu verhindern (Rainbowtables), wird ein IV als Zufallsvariable der Berechnung hinzugefügt.
Garbage collector	Gibt nicht mehr benötigte Speicherbereiche eines Programms frei.

9.3 Abbildungsverzeichnis

Abbildung 1: AES Block Verschlüsselung	13
Abbildung 2: ECB Verschlüsselungsmodus	13
Abbildung 3: CBC Verschlüsselungsmodus	13
Abbildung 4: RSA Verschlüsselung	14
Abbildung 5: Inhalt einer ODT-Datei	15
Abbildung 6: Inhalt einer DOCX-Datei	16
Abbildung 7: Funktionsweise von XSLT	16
Abbildung 8: Systemüberblick	23
Abbildung 9: Domänenmodell SecureExam	23
Abbildung 10: Beispiel eines Exception rethrowings	24
Abbildung 11: Multi-User AES Verschlüsselung	25
Abbildung 12: Multi-User AES Entschlüsselung	26
Abbildung 13: Hashing Ablauf	27
Abbildung 14: Manipulation der internen Uhr detektieren	29
Abbildung 15: Vergleich der internen Uhr mit der Systemuhr	29
Abbildung 16: UI Design Idee	30
Abbildung 17: Login-Skizze	31
Abbildung 18: Skizze Paging Mode	31
Abbildung 19: Skizze Scroll Mode	32
Abbildung 20: Kryptographie Import in C#	34
Abbildung 21: AES mit Using-Scopes	34
Abbildung 22: CryptoJS AES decryption mit custom IV & Key	35
Abbildung 23: SHA256 Hashing in C#	35
Abbildung 24: SHA256 Berechnung mit CryptoJS	36
Abbildung 25: Beispiel einer Prüfungskonfigurationsdatei	36
Abbildung 26: Links Prüfungshinweise mit HTML, Rechts die Darstellung im Browser	37
Abbildung 27: Struktur von SecureExam.xml	37
Abbildung 28: XML-Auszug mit einer Frage	38
Abbildung 29: Auszug XSLT	38
Abbildung 30: Beispiel von Einträgen im Studenten Import File	39
Abbildung 31: Auszug aus der Skeleton-Datei	39
Abbildung 32: Das Ersetzen der Platzhalter als Auszug aus der C#-Exportfunktion	39
Abbildung 33: Beispiel von exportierten Studenten Login Daten	40
Abbildung 34: Konfigurationsparameter zum Handling des CPU Lags (Parameter in ms)	41
Abbildung 35: DateCompate Funktion und Methode zur Kompensation des CPU Lag	41
Abbildung 36: Dynamische Einbindung eines neuen Bildes zur Prüfung der Internetverbindung	42
Abbildung 37: Authentifizierungsansicht	42
Abbildung 38: Prüfung im Page-Mode	43
Abbildung 39: Prüfung im Scroll-Mode blendet die Hinweise aus	43
Abbildung 40: Add Fakes Assembly	44
Abbildung 41: C# Fakes	44
Abbildung 42: Konsolenausgabe umleiten in C#	45
Abbildung 43: TestExplorer mit den Unit Tests	45

Abbildung 44: Auszug der Code Coverage Analyzer Ausgabe.....	46
Abbildung 45: Prüfung auf PC mit Scroll-Funktion.....	46
Abbildung 46: Prüfung im Paging-Mode auf Tablet mit elnk Bildschirm	47
Abbildung 47: Prüfung im Paging-Mode auf Tablet mit LCD-Bildschirm	48

9.4 Tabellenverzeichnis

Tabelle 1: Vergleich AES und RSA.....	24
Tabelle 2: Vergleich SHA3 und PBKDF2	26
Tabelle 3: Vergleich ODT und Word	27
Tabelle 4: Vergleich HTML und PDF	28
Tabelle 5: JavaScript Obfuscation.....	33
Tabelle 6: CPU Lag Analyse (Testsystem: Intel I7-4500U Windows 8.1 64Bit mit Chrome 40.0.2214.28 beta-m (64-bit))	40

10 Anhang

10.1 Projektmanagement

10.1.1 Zeitplan

Grobe Zeitplanung Projekt: Sichere Tests auf E-Book Readern														
	Planung		Evaluation		Entwicklungsphase					Test	Abschluss			
	KW38	KW39	KW40	KW41	KW42	KW43	KW44	KW45	KW46	KW47	KW48	KW49	KW50	KW51
Überblick verschaffen														
Eingesetzte Technologie bestimmen														
PrüfungsGeneratorTool entwickeln														
JS decryption code entwickeln														
JS Manipulationsschutzmechanismen entwickeln														
Gerät evaluieren														
Gerät beschaffen														
Testen														
Code Freeze														
Pufferzeit														
Dokumentation														

10.1.2 Sitzungsprotokolle

10.1.2.1 Sitzung 1:

Datum: 15. September 2014

Anwesend: Karl Rege, Simon Lukes, Daniel Jampen

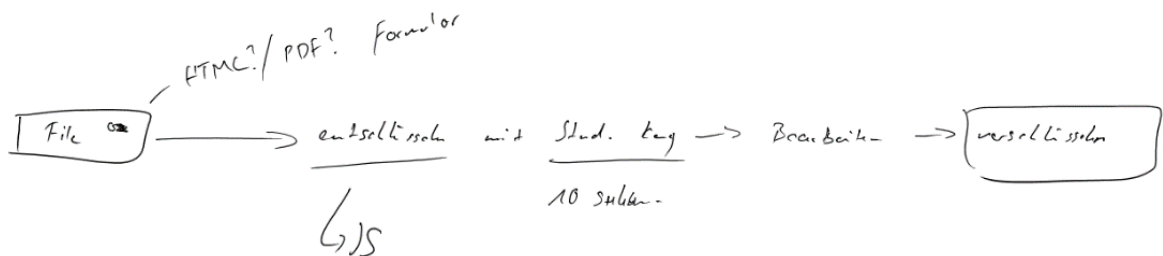
Abwesend: -

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

10.1.2.2 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)

10.1.2.3 Sitzung3

Datum: 29. Oktober 2014

Anwesend: Karl Rege, Simon Lukes, Daniel Jampen

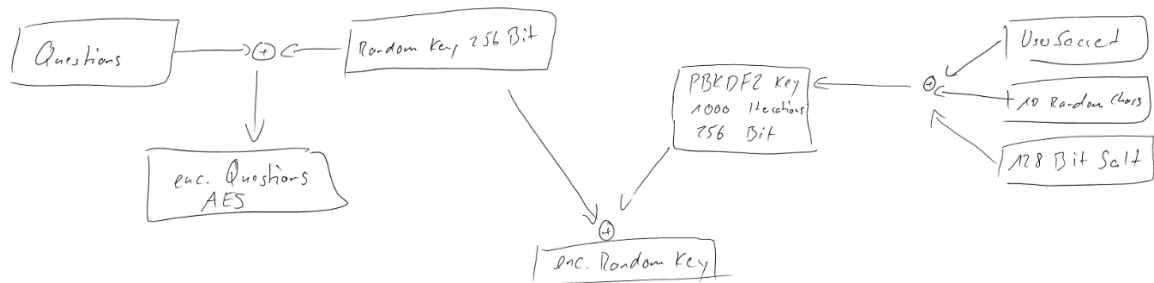
Abwesend: -

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)

10.1.2.4 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

10.1.2.5 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 besser 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?

10.1.2.6 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

10.1.2.7 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:
 - o Prüfungsfenster
 - o Öffnungszeit und Anzahl Öffnungen loggen und in Export schreiben
 - o AutoSave Funktion
 - o TabFocus lost
 - o InternetConnection check

10.1.2.8 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
- Prüfungszeitfenster => 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:

- Prüfungszeitraum und Prüfungsdauer sollten mit Fragen verschlüsselt werden. Manipulationschutz
- 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

10.1.2.9 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 ausschalten
- 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

10.1.2.10 Sitzung10

Datum: 17. November 2014

Anwesend: Karl Rege, Simon Lukes, Daniel Jampen

Abwesend: -

Erledigte Arbeiten:

- Security-Features ein- und ausschaltbar 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.

10.1.2.11 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

10.1.2.12 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

10.1.2.13 Sitzung13

Datum: 8. Dezember 2014

Anwesend: Karl Rege, Simon Lukes, Daniel Jampen

Abwesend: -

Erledigte Arbeiten:

- Dokumentieren

Besprochene Tendenzen:

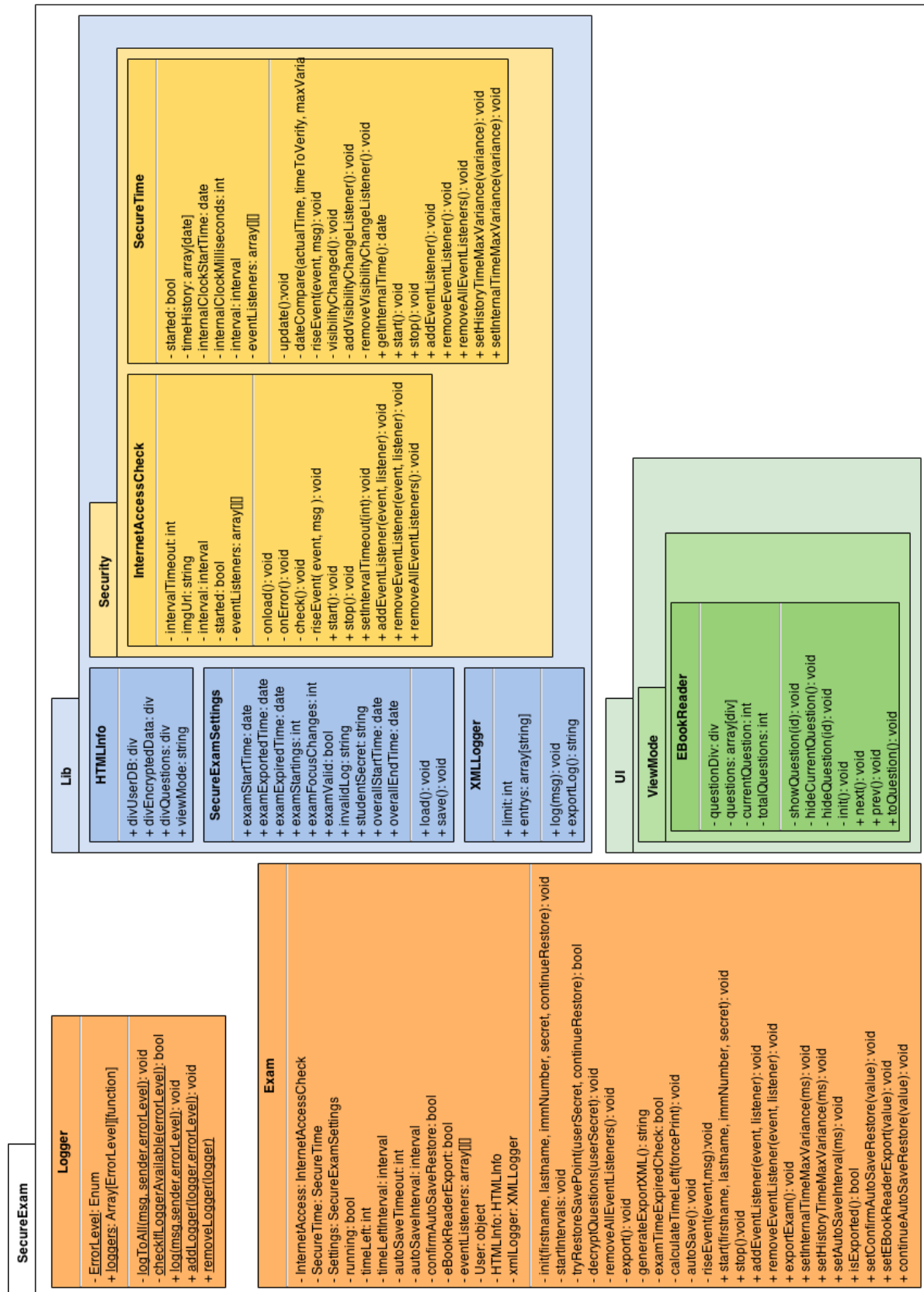
- Finale Fragen zu Dokumentation geklärt
- Letzte Sitzung abgesagt, da unnötig.

Zu erledigende Arbeiten:

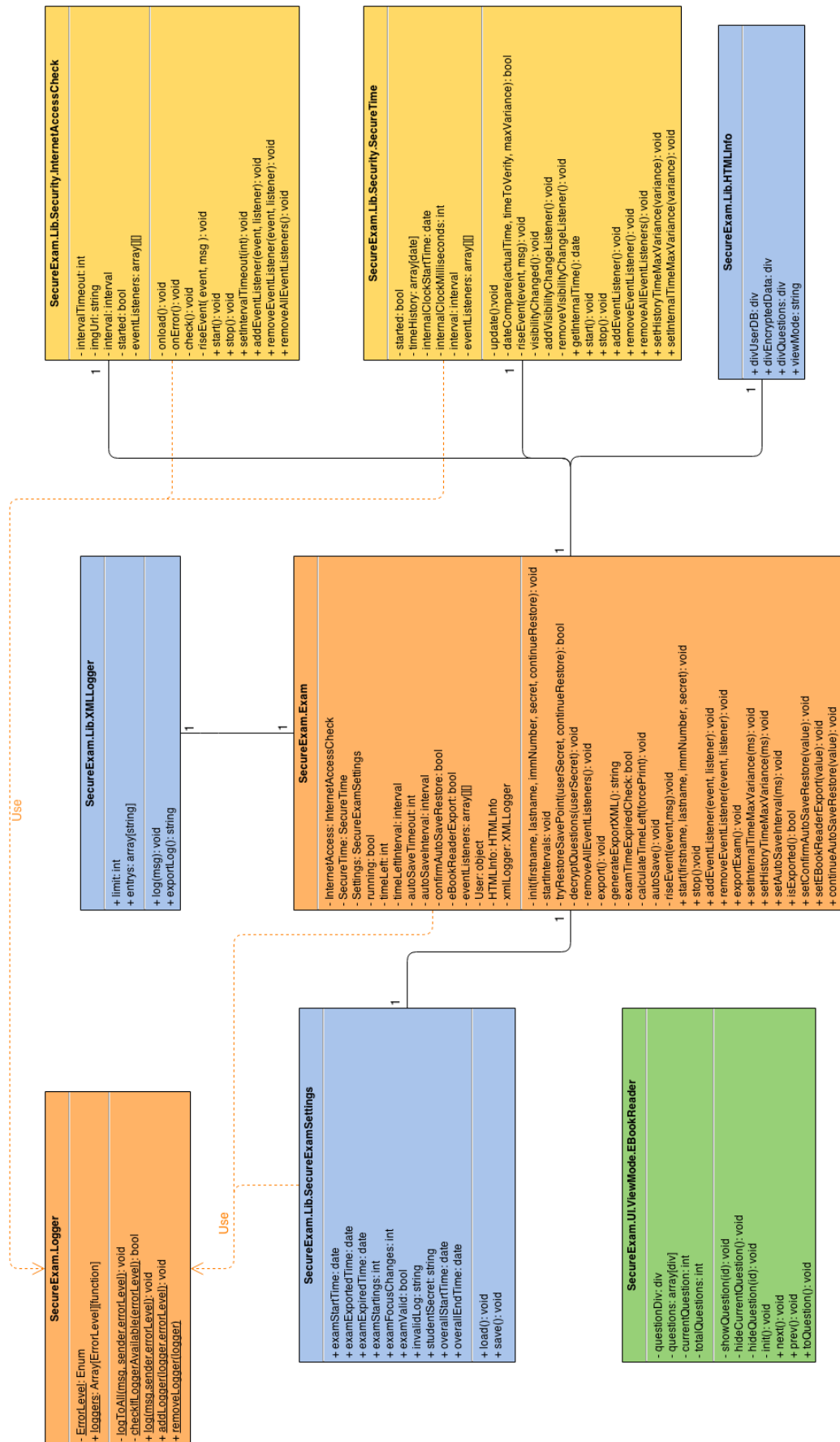
- Doku abschliessen

10.1.4 SecureExam JavaScript Library

10.1.4.1 Overview



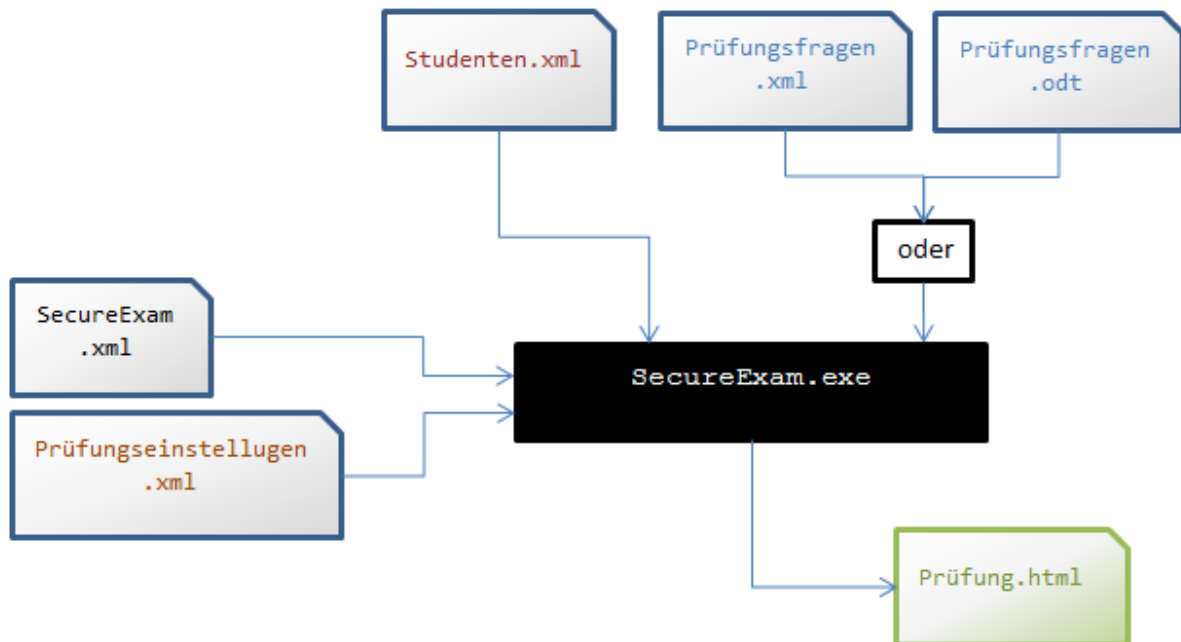
10.1.4.2 Klassendiagramm



10.2 Bedienungsanleitung

10.2.1 Konsolenapplikation

10.2.1.1 Überblick



Die Applikation SecureExam ist eine Konsolenapplikation, welche verschiedene Parameter verlangt. Eine minimale Konfiguration zum Generieren einer Prüfung lautet folgendermassen:

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

Mit diesen Parametern werden nur die Ein- und Ausgabedateien angegeben, die gewählten Formate entsprechen alle den Standardwerten.

10.2.1.2 Konsolenparameter

Folgende Parameter der Konsolenapplikation sind obligatorisch und müssen 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]**

Die zur Verfügung stehenden Typen sind mittels eines Aufrufs der Applikation ohne Parameter ersichtlich.

10.2.1.3 Import

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

10.2.1.3.1 SecureExam.xml

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

```
<data>
  <NumberOfRandomCharsInStudentSecret>10</NumberOfRandomCharsInStudentSecret>
  <AESSettings>
    <keyLength>256</keyLength>
    <ivLength>128</ivLength>
  </AESSettings>
  <SHA256Settings>
    <iterations>100000</iterations>
    <saltLength>256</saltLength>
  </SHA256Settings>
</data>
```

Alle Elemente in diesem File sind obligatorisch und müssen angegeben werden. Sie müssen allerdings nur verändert werden, wenn ein anderes Verhalten des Prüfungsgenerators gewünscht wird.

10.2.1.3.1.1 NumberOfRandomCharsInStudentSecret

Dieses Element dient dazu, die Länge des Passwortes zu definieren.

10.2.1.3.1.2 AESSettings

Die Einstellung <keyLength> definiert die Länge des Master-Schlüssels, welcher für die Ver- und Entschlüsselung verwendet wird. <ivLength> definiert die Länge des Initialisierungsvektors. Es ist empfohlen, diese Einstellung auf den Standardwerten zu belassen.

10.2.1.3.1.3 SHA256Settings

Mithilfe von <iterations> wird die Anzahl von Hash-Durchgängen festgelegt. Je grösser die Anzahl Iterationen ist, desto länger dauert das Hashing und somit 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.

10.2.1.3.2 Prüfungseinstellungen.xml

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

```
<settings>
  <subject>ADS</subject>
  <examTitle>Algorithmen und Datenstrukturen</examTitle>
  <examNotes>Keine Hilfsmittel erlaubt</examNotes>
  <examDate>1.12.2014</examDate>
  <startTime>14:00</startTime>
  <endTime>16:00</endTime>
  <duration>45</duration>
  <internetAllowed>false</internetAllowed>
  <tabChangeAllowed>true</tabChangeAllowed>
  <historyTimeMaxVariance>5000</historyTimeMaxVariance>
  <internalTimeMaxVariance>5000</internalTimeMaxVariance>
  <confirmAutosaveRestore>true</confirmAutosaveRestore>
  <ebookreaderExport>true</ebookreaderExport>
  <viewMode>scroll</viewMode>
</settings>
```

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. HTML ist zur Formatierung der Hinweise erlaubt.	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 während der Prüfung der Tab des Browsers gewechselt werden darf. Mögliche Antworten sind „true“ oder „false“. Als Standard ist „false“ definiert.	Nein
<historyTimeMaxVariance>	Setzt die maximale Abweichung der letzten zehn Zeitmessungen. Werte unter 20ms werden nicht unterstützt.	Ja
<internalTimeMaxVariance>	Setzt die maximale Abweichung der eigenen Zeitrech-	Ja

	nung zur Uhr des Systems. Damit wird festgestellt, ob die Systemzeit versucht wird zu manipulieren. Werte unter 20ms werden nicht unterstützt.	
<confirmAutosaveRestore>	Definiert, ob die automatisch gesicherten Daten automatisch wiederhergestellt werden sollen oder ob der Benutzer die Möglichkeit erhält darüber zu entscheiden. Falls der Benutzer dies ablehnt, wird ihm eine neue Prüfung erstellt, jedoch wird die verbliebene Prüfungszeit nicht zurückgesetzt! Mögliche Antworten sind „true“ oder „false“. Als Standard ist „false“ definiert.	Nein
<ebookreaderExport>	eBook Reader haben möglicherweise keine Downloadunterstützung deshalb kann festgelegt werden, ob der Export der Prüfung in die LocalDB des Browsers gespeichert wird oder ob eine Datei zum Download bereitgestellt werden soll. Mögliche Antworten sind „true“ (LocalDB) oder „false“ (Download). 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

10.2.1.3.3 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.

```
<data>
  <professor>
    <name>Rege</name>
    <vorname>Karl</vorname>
  </professor>
  <students>
    <student>
      <name>Lukes</name>
      <vorname>Simon</vorname>
      <number>S12198347</number>
    </student>
  </students>
</data>
```


10.2.1.3.4 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.

10.2.1.3.4.1 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.

```
<questions>
  <question>
    <legend>Frage 1. Welche der folgenden Operationen kommen in einem Stack vor?</legend>
    <input type="checkbox" value="enter" />
    <input type="checkbox" value="top" />
    <input type="checkbox" value="empty" />
    <input type="checkbox" value="front" />
    <input type="checkbox" value="push" />
  </question>
</questions>
```

Für jede Frage 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 Checkboxes 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. Dieses funktioniert genau gleich wie normale HTML Placeholders.

10.2.1.3.4.2 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.

10.2.1.4 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.

10.2.2 Generierte Prüfung

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

10.2.2.1 Startseite

Gleich nach dem Starten des Programms wird folgende Seite angezeigt.

ZHAW School of Engineering

ADS: Algorithmen und Datenstrukturen

Prüfungshinweise:

- Es sind mehrere Antworten möglich
- Rechner erlaubt!

Bitte geben Sie die geforderten Informationen ein um die Prüfung starten zu können! Viel Glück!

Vorname:

Nachname:

Immatrikulationsnummer:

Passwort:

Prüfung starten!

10.2.2.2 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.

Bitte geben Sie die geforderten Informationen ein um die Prüfung starten zu können! Viel Glück!

Vorname:

Nachname:

Immatrikulationsnummer:

Passwort:

Nach dem Bestätigen der Eingabe durch einen Klick auf „Prüfung starten“, wird die Prüfung entschlüsselt und angezeigt. Falls nicht alle Felder ausgefüllt werden, wird folgende Meldung angezeigt:

JavaScript-Warnmeldung

Bitte alle Felder richtig ausfüllen!

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

10.2.2.3 Prüfungsfenster

ZHAW School of Engineering Verbliebene Zeit: 18:00

ADS: Algorithmen und Datenstrukturen

Vorname: Simon Nachname: Lukes Immatrikulationsnummer: S12198347

Prüfungshinweise:

- Es sind mehrere Antworten möglich
- Rechner erlaubt!

Frage 1. Welche der folgenden Operationen kommen in einem Stack vor?

☐ enter

☐ top

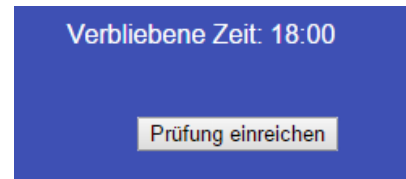
☐ empty

☐ front

☐ push

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.



10.2.2.4 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:



10.3 Weiteres

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