

**Projektarbeit (Informatik)**

Sichere eTest auf eBook Readern

|  |  |  |
| --- | --- | --- |
| **Autoren** |  | Simon Lukes  Daniel Jampen |
| **Hauptbetreuung** |  | Karl Rege |
| **Datum** |  | 09.12.2014 |



Zusammenfassung

Abstract… english

Inhaltsverzeichnis

[1 Einleitung 9](#_Toc405888559)

[1.1 Ausgangslage 9](#_Toc405888560)

[1.2 Problemstellung 9](#_Toc405888561)

[1.3 Zielsetzung / Aufgabenstellung / Anforderungen 9](#_Toc405888562)

[1.3.1 Muss Funktionen 9](#_Toc405888563)

[1.3.2 Wunsch Funktionen 9](#_Toc405888564)

[2 Grundlagen 10](#_Toc405888565)

[2.1 Kryptographie 10](#_Toc405888566)

[2.1.1 Verschlüsselung 10](#_Toc405888567)

[2.1.2 Passwort Hashing 11](#_Toc405888568)

[2.2 Dateiformate 12](#_Toc405888569)

[2.2.1 INI 12](#_Toc405888570)

[2.2.2 XML 12](#_Toc405888571)

[2.2.3 Open Office 12](#_Toc405888572)

[2.2.4 Microsoft Word 13](#_Toc405888573)

[2.2.5 XSLT 13](#_Toc405888574)

[2.2.6 PDF 13](#_Toc405888575)

[2.2.7 HTML mit Javascript 14](#_Toc405888576)

[2.3 Digitale Manipulationsmöglichkeiten 14](#_Toc405888577)

[2.3.1 Korrekte Antwort auslesen 14](#_Toc405888578)

[2.3.2 Zeit 14](#_Toc405888579)

[2.3.3 Internet Zugriff 14](#_Toc405888580)

[2.3.4 Prüfungsinterne Kommunikation 14](#_Toc405888581)

[2.4 eBook Reader 15](#_Toc405888582)

[2.4.1 Einschränkungen 15](#_Toc405888583)

[3 Analyse 16](#_Toc405888584)

[3.1 Anwendungsfälle 16](#_Toc405888585)

[3.1.1 Übersicht 16](#_Toc405888586)

[3.1.2 Akteure 16](#_Toc405888587)

[3.1.3 AF01 Prüfung erstellen 16](#_Toc405888588)

[3.1.4 AF02 Prüfung absolvieren 16](#_Toc405888589)

[3.1.5 AF02 Abgelegte Prüfung auslesen 16](#_Toc405888590)

[3.2 Nichtfunktionale Anforderungen 16](#_Toc405888591)

[3.2.1 NFA01 Prüfung ist einfach bedienbar 16](#_Toc405888592)

[3.2.2 NFA02 Export erfolgt transparent für den Probanden 16](#_Toc405888593)

[3.2.3 NFA03 Anmeldevorgang dauert weniger als drei Sekunden 16](#_Toc405888594)

[4 Konzept 17](#_Toc405888595)

[4.1 Systemüberblick 17](#_Toc405888596)

[4.1.1 Domänenmodell 18](#_Toc405888597)

[4.2 C# Errorhandling 18](#_Toc405888598)

[4.3 Codedokumentation 19](#_Toc405888599)

[4.3.1 C# 19](#_Toc405888600)

[4.3.2 JavaScript 19](#_Toc405888601)

[4.4 Export 19](#_Toc405888602)

[4.4.1 Evaluationsmatrix 19](#_Toc405888603)

[4.4.2 Entscheid 19](#_Toc405888604)

[4.5 Kryptographie 20](#_Toc405888605)

[4.5.1 Verschlüsselung 20](#_Toc405888606)

[4.5.2 Passwort hashing 22](#_Toc405888607)

[4.6 Konfigurationsdateien 22](#_Toc405888608)

[4.6.1 Entscheid 22](#_Toc405888609)

[4.6.2 Konzept 23](#_Toc405888610)

[4.7 Import 23](#_Toc405888611)

[4.7.1 Evaluationsmatrix 23](#_Toc405888612)

[4.7.2 Entscheid 23](#_Toc405888613)

[4.8 Digitale Manipulationsmöglichkeiten 23](#_Toc405888614)

[4.8.1 Zeit 23](#_Toc405888615)

[4.8.2 Internet Zugriff 24](#_Toc405888616)

[4.8.3 Prüfungsinterne Kommunikation 24](#_Toc405888617)

[4.9 UI-Design 25](#_Toc405888618)

[4.9.1 Idee 25](#_Toc405888619)

[4.9.2 Login 26](#_Toc405888620)

[4.9.3 Prüfung 26](#_Toc405888621)

[4.10 JavaScript 27](#_Toc405888622)

[4.10.1 SecureExam Library 27](#_Toc405888623)

[4.10.2 CryptoJS Library 27](#_Toc405888624)

[4.10.3 FileSaver.js Library 28](#_Toc405888625)

[4.10.4 Code Obfuscation 28](#_Toc405888626)

[5 Umsetzung 29](#_Toc405888627)

[5.1 Export 29](#_Toc405888628)

[5.2 Kryptographie 29](#_Toc405888629)

[5.2.1 Verschlüsselung 29](#_Toc405888630)

[5.2.2 Entschlüsselung 31](#_Toc405888631)

[5.2.3 Passwort hashing 31](#_Toc405888632)

[5.3 Konfigurationsdateien 32](#_Toc405888633)

[5.3.1 Prüfungskonfigurationsdatei 32](#_Toc405888634)

[5.3.2 SecureExam.xml 33](#_Toc405888635)

[5.4 Import 33](#_Toc405888636)

[5.4.1 XML 34](#_Toc405888637)

[5.4.2 Open Office 34](#_Toc405888638)

[5.5 Digitale Manipulationsmöglichkeiten 35](#_Toc405888639)

[5.5.1 Zeit 35](#_Toc405888640)

[5.5.2 Internet Zugriff 37](#_Toc405888641)

[5.6 UI-Design 37](#_Toc405888642)

[5.6.1 Farben 37](#_Toc405888643)

[5.6.2 Authentifizierungsansicht 37](#_Toc405888644)

[5.6.3 Page-Mode 38](#_Toc405888645)

[5.6.4 Scroll-Mode 38](#_Toc405888646)

[6 Testing 39](#_Toc405888647)

[6.1 Konzept 39](#_Toc405888648)

[6.1.1 C# 39](#_Toc405888649)

[6.1.2 HTML / JavaScript 39](#_Toc405888650)

[6.2 C# Code 39](#_Toc405888651)

[6.3 HTML / JavaScript Funktionalität 39](#_Toc405888652)

[7 Resultate 39](#_Toc405888653)

[8 Diskussion und Ausblick 39](#_Toc405888654)

[9 Verzeichnisse 39](#_Toc405888655)

[9.1 Literaturverzeichnis 39](#_Toc405888656)

[9.2 Glossar 39](#_Toc405888657)

[9.3 Abbildungsverzeichnis 39](#_Toc405888658)

[9.4 Tabellenverzeichnis 40](#_Toc405888659)

[10 Anhang 41](#_Toc405888660)

[10.1 Projektmanagement 41](#_Toc405888661)

[10.1.1 Zeitplan 41](#_Toc405888662)

[10.1.2 Sitzungsprotokolle 41](#_Toc405888663)

[10.2 C# Klassendiagramm 41](#_Toc405888664)

[10.3 Bedienungsanleitung 41](#_Toc405888665)

[10.3.1 Konsolenapplikation 41](#_Toc405888666)

[10.3.2 OpenOffice Import 41](#_Toc405888667)

[10.3.3 Generierte Prüfung 41](#_Toc405888668)

[10.4 Weiteres 41](#_Toc405888669)

[10.4.1 CD mit dem vollständigen Bericht als pdf-File und dem SourceCode 41](#_Toc405888670)

# Einleitung

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

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

## Ausgangslage

* Nennt bestehende Arbeiten/Literatur zum Thema  Literaturrecherche
* Stand der Technik: Bisherige Lösungen des Problems und deren Grenzen

### Problemstellung

Diverse Probleme wurden digitalen Prüfungen bisher zum Verhängnis. Nachfolgend werden die Grundproblematiken aufgeführt und näher beschrieben:

**Manipulationen:**

Fu

**Prüfungsdateien werden gestohlen:**

Simon

**Unzuverlässige Infrastruktur:**

I MACHE

**Kommunikationsfähigkeit der Geräte:**

NÜMME!!

## Aufgabenstellung

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

## Anforderungen

### Muss Funktionen

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

### Wunsch Funktionen

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

## Zielsetzung

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

## Zielpublikum

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

**Prüfungsgenerator:**

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

**Prüfung:**

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

# Grundlagen

## Kryptographie

### Verschlüsselung

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

#### AES

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



Abbildung : AES Block Verschlüsselung

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

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

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

#### RSA

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



Abbildung : 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.

### Passwort Hashing

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

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

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

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

#### SHA3

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

#### PBKDF2

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

PBKDF2 wendet auf den Input einen Hashalgorithmus mit Salting und anschliessendem Chaining an.

## Dateiformate

### INI

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

### XML

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

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

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

### Open Office

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



Abbildung : Inhalt einer ODT-Datei

### Microsoft Word

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



Abbildung : Inhalt einer DOCX-Datei

### XSLT

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

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

Abbildung : Funktionsweise von XSLT

### PDF

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

### HTML mit Javascript

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

## Digitale Manipulationsmöglichkeiten

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

### Korrekte Antwort auslesen

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

### Zeit

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

#### Manipulation der geräteinternen Uhr

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

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

#### Verlangsamung der Zeit

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

### Internet Zugriff

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

### Prüfungsinterne Kommunikation

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

#### Wireless LAN

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

#### Bluetooth Modul

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

## eBook Reader

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

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

### Einschränkungen

#### Betriebssystem

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

#### CPU / GPU Geschwindigkeit

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

#### Displaytechnik

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

# Analyse

## Anwendungsfälle

Sicht? Anwender!

### Übersicht

### Akteure

#### Professor

#### Student

### AF01 Prüfung erstellen

### AF02 Prüfung absolvieren

### AF02 Abgelegte Prüfung auslesen

## Nichtfunktionale Anforderungen

### NFA01 Prüfung ist einfach bedienbar

### NFA02 Export erfolgt transparent für den Probanden

### NFA03 Anmeldevorgang dauert weniger als drei Sekunden

Auf grund vom entschlüsseln.. blablabli

# Konzept

## Systemüberblick



Abbildung : Systemüberblick

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

### Domänenmodell



Abbildung :Domänenmodell SecureExam

## C# Errorhandling

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



Abbildung : Beispiel eines Exception rethrowings

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

## Codedokumentation

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

### C#

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

### JavaScript

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

## Export

### Evaluationsmatrix



Tabelle : Vergleich HTML und PDF

### Entscheid

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

## Kryptographie

### Verschlüsselung

#### Evaluationsmatrix



Tabelle : Vergleich AES und RSA

#### Entscheid

Wie bei der Evaluationsmatrix(vgl. 4.3.1.1) ersichtlich, hat AES gegenüber RSA einen grossen Vorteil in Bezug auf die „einfache Keyeingabe“. Bei AES kann man ein normales Passwort mit anschliessendem Hashing verwenden, bei RSA müsste ein komplettes Zertifikat korrekt abgetippt werden. Deshalb wird AES als Verschlüsselungsverfahren eingesetzt.

#### Konzept

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

##### Verschlüsselung



Abbildung : Multi-User AES Verschlüsselung

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

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

Diese werden anschliessend n-Mal mit gehasht (vgl. 4.3.2, Passwort hashing). Mit dem resultierenden Hash (256 Bit) und einem zufälligen IV (Initialisierungsvektor) wird nun der Masterkey mittels AES verschlüsselt. Der Masterkey setzt sich aus einer zufälligen, 256 Bit langen Zeichenkette zusammen. Dieser wird dazu verwendet, um die Prüfungsdaten (Fragen, Prüfungszeitraum und Prüfungsdauer) zu verschlüsseln.

##### Entschlüsselung



Abbildung : Multi-User AES Entschlüsselung

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

* Vorname
* Nachname
* Immatrikulationsnummer
* n zufällige Zeichen

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

### Passwort hashing

#### Evaluationsmatrix



Tabelle : Vergleich SHA3 und PBKDF2

#### Entscheid

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

#### Hashing Konzept

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



Abbildung : Hashing Ablauf

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

## Konfigurationsdateien

### Entscheid

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

### Konzept

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

## Import

### Evaluationsmatrix



Tabelle : Vergleich ODT und Word

### Entscheid

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

## Digitale Manipulationsmöglichkeiten

### Zeit

#### Manipulation der geräteinternen Uhr

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



Abbildung : Manipulation der internen Uhr detektieren

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

#### Verlangsamung der Zeit

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



Abbildung : Vergleich der internen Uhr mit der Systemuhr

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

### Internet Zugriff

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

### Prüfungsinterne Kommunikation

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

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

## UI-Design

### Idee

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



Abbildung : UI Design Idee

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

### Login



Abbildung : Login-Skizze

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

* Vorname
* Name
* Immatrikulationsnummer
* Passwort

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

### Prüfung

#### Page-Mode



Abbildung : Skizze Paging Mode

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

#### Scroll-Mode

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

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

Abbildung : Skizze Scroll Mode

## JavaScript

### SecureExam Library

[Klassendiagramm]

### CryptoJS Library

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

* SHA3
* AES
* Bit / Byte Datenstrukturen

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

### FileSaver.js Library

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

### Code Obfuscation

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

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

Tabelle : JavaScript Obfuscation

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

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

# Umsetzung

## Export

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



Abbildung : Auszug aus der Skeleton-Datei

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



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

## Kryptographie

### Verschlüsselung

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



Abbildung : Kryptographie Import in C#

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



Abbildung : AES mit Using-Scopes

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

### Entschlüsselung

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



Abbildung : CryptoJS AES decryption mit custom IV & Key

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

### Passwort hashing

#### SecureExam.exe

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



Abbildung : SHA256 Hashing in C#

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

#### JavaScript

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



Abbildung : SHA256 Berechnung mit CryptoJS

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

## Konfigurationsdateien

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

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

### Prüfungskonfigurationsdatei

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



Abbildung : Beispiel einer Prüfungskonfigurationsdatei

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

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

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

#### Prüfungshinweise

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



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

### SecureExam.xml

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

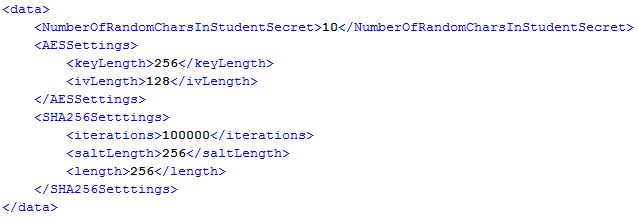


Abbildung : Struktur von SecureExam.xml

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 28 ersichtlich ist, werden sowohl die Anzahl von Zufallszahlen für das Erstellen des Passworts definiert als auch Einstellungen zum Ver- und Entschlüsseln der Prüfung mittels AES und SHA256 vorgenommen. Die Verwendung dieser Kryptografie-Parameter werden im Abschnitt 4.2 Kryptographie und in der Bedienungsanleitung genauer erläutert.

## Import

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

### XML

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



Abbildung : XML-Auszug mit einer Frage

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

Auszug aus Code?

### Open Office

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



Abbildung : Auszug XSLT

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

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

## Digitale Manipulationsmöglichkeiten

### Zeit

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

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



Tabelle : CPU Lag Analyse

Aus der Analyse kann man folgende Schlüsse ziehen:

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

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

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

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



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

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



Abbildung : DateCompate Funktion und Methode zur Kompensation des CPU Lag

### Internet Zugriff

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

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

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



Abbildung : Dynamische Einbindung eines neuen Bildes zur Prüfung der Internetverbindung

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

## UI-Design

### Farben

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

### Authentifizierungsansicht



Abbildung : Authentifizierungsansicht

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

### Page-Mode

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



Abbildung : Prüfung im Page-Mode

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

### Scroll-Mode

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



Abbildung : Prüfung im Scroll-Mode

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

# Testing

## Konzept

### C#

### HTML / JavaScript

## C# Code

## HTML / JavaScript Funktionalität

# Resultate

Lauffähig auf eBook Reader und Tablet => bilder bla bla

# Diskussion und Ausblick

* Bespricht die erzielten Ergebnisse bezüglich ihrer Erwartbarkeit, Aussagekraft und Relevanz
* Interpretation und Validierung der Resultate
* Rückblick auf Aufgabenstellung, erreicht bzw. nicht erreicht
* Legt dar, wie an die Resultate (konkret vom Industriepartner oder weiteren Forschungsarbeiten; allgemein) angeschlossen werden kann; legt dar, welche Chancen die Resultate bieten
* Ausblick

# Verzeichnisse

## Literaturverzeichnis

## Glossar

## Abbildungsverzeichnis

[Abbildung 1: AES Block Verschlüsselung 10](#_Toc405845844)

[Abbildung 2: ECB Verschlüsselungsmodus 10](#_Toc405845845)

[Abbildung 3: CBC Verschlüsselungsmodus 10](#_Toc405845846)

[Abbildung 4: RSA Verschlüsselung 11](#_Toc405845847)

[Abbildung 5: Inhalt einer ODT-Datei 12](#_Toc405845848)

[Abbildung 6: Inhalt eines Word Dokuments 13](#_Toc405845849)

[Abbildung 7: Systemüberblick 17](#_Toc405845850)

[Abbildung 8:Domänenmodell SecureExam 18](#_Toc405845851)

[Abbildung 9: Beispiel eines Exception rethrowings 18](#_Toc405845852)

[Abbildung 10: Multi-User AES Verschlüsselung 20](#_Toc405845853)

[Abbildung 11: Multi-User AES Entschlüsselung 21](#_Toc405845854)

[Abbildung 12: Hashing Ablauf 22](#_Toc405845855)

[Abbildung 13: Manipulation der internen Uhr detektieren 23](#_Toc405845856)

[Abbildung 14: Vergleich der internen Uhr mit der Systemuhr 24](#_Toc405845857)

[Abbildung 15: UI Design Idee 25](#_Toc405845858)

[Abbildung 16: Login-Skizze 26](#_Toc405845859)

[Abbildung 17: Skizze Paging Mode 26](#_Toc405845860)

[Abbildung 18: Skizze Scroll Mode 27](#_Toc405845861)

[Abbildung 19: Auszug aus der Skeleton-Datei 29](#_Toc405845862)

[Abbildung 20: Das Ersetzen der Platzhalter als Auszug aus der C#-Exportfunktion 29](#_Toc405845863)

[Abbildung 21: Kryptographie Import in C# 29](#_Toc405845864)

[Abbildung 22: AES mit Using-Scopes 30](#_Toc405845865)

[Abbildung 23: CryptoJS AES decryption mit custom IV & Key 31](#_Toc405845866)

[Abbildung 24: SHA256 Hashing in C# 31](#_Toc405845867)

[Abbildung 25: SHA256 Berechnung mit CryptoJS 32](#_Toc405845868)

[Abbildung 26: Beispiel einer Prüfungskonfigurationsdatei 32](#_Toc405845869)

[Abbildung 27: Links Prüfungshinweise mit HTML, Rechts die Darstellung im Browser 33](#_Toc405845870)

[Abbildung 28: Struktur von SecureExam.xml 33](#_Toc405845871)

[Abbildung 29: XML-Auszug mit einer Frage 34](#_Toc405845872)

[Abbildung 30: Auszug XSLT 34](#_Toc405845873)

[Abbildung 31: Konfigurationsparameter zum Handling des CPU Lags (Parameter in ms) 36](#_Toc405845874)

[Abbildung 32: DateCompate Funktion und Methode zur Kompensation des CPU Lag 36](#_Toc405845875)

[Abbildung 33: Dynamische Einbindung eines neuen Bildes zur Prüfung der Internetverbindung 37](#_Toc405845876)

[Abbildung 34: Authentifizierungsansicht 37](#_Toc405845877)

[Abbildung 35: Prüfung im Page-Mode 38](#_Toc405845878)

[Abbildung 36: Prüfung im Scroll-Mode 38](#_Toc405845879)

## Tabellenverzeichnis

[Tabelle 1: Vergleich HTML und PDF 19](#_Toc405845838)

[Tabelle 2: Vergleich AES und RSA 20](#_Toc405845839)

[Tabelle 3: Vergleich SHA3 und PBKDF2 22](#_Toc405845840)

[Tabelle 4: Vergleich ODT und Word 23](#_Toc405845841)

[Tabelle 5: JavaScript Obfuscation 28](#_Toc405845842)

[Tabelle 6: CPU Lag Analyse 35](#_Toc405845843)

# Anhang

## Projektmanagement

### Zeitplan



### Sitzungsprotokolle

## C# Klassendiagramm

[Bild]

Modularität bla bla bla

## Bedienungsanleitung

### Konsolenapplikation

### OpenOffice Import

### Generierte Prüfung

## Weiteres

### CD mit dem vollständigen Bericht als pdf-File und dem SourceCode