

**Projektarbeit (Informatik)**

Sichere eTest auf eBook Readern

|  |  |  |
| --- | --- | --- |
| **Autoren** |  | Simon Lukes  Daniel Jampen |
| **Hauptbetreuung** |  | Karl Rege |
| **Datum** |  | 07.12.2014 |



Zusammenfassung

Abstract… english

Inhaltsverzeichnis

[1 Einleitung 9](#_Toc405734470)

[1.1 Ausgangslage 9](#_Toc405734471)

[1.2 Problemstellung 9](#_Toc405734472)

[1.3 Zielsetzung / Aufgabenstellung / Anforderungen 9](#_Toc405734473)

[1.3.1 Muss Funktionen 9](#_Toc405734474)

[1.3.2 Wunsch Funktionen 9](#_Toc405734475)

[2 Grundlagen 10](#_Toc405734476)

[2.1 Exportformat 10](#_Toc405734477)

[2.1.1 PDF 10](#_Toc405734478)

[2.1.2 HTML mit Javascript 10](#_Toc405734479)

[2.2 Kryptographie 10](#_Toc405734480)

[2.2.1 Verschlüsselung 10](#_Toc405734481)

[2.2.2 Passwort Hashing 11](#_Toc405734482)

[2.3 Konfigurationsdateien 12](#_Toc405734483)

[2.3.1 INI 12](#_Toc405734484)

[2.3.2 XML 12](#_Toc405734485)

[2.4 Import 12](#_Toc405734486)

[2.4.1 XML 12](#_Toc405734487)

[2.4.2 Open Office 13](#_Toc405734488)

[2.4.3 Microsoft Word 13](#_Toc405734489)

[2.4.4 HTML 13](#_Toc405734490)

[2.5 Digitale Manipulationsmöglichkeiten 13](#_Toc405734491)

[2.5.1 Korrekte Antwort auslesen 13](#_Toc405734492)

[2.5.2 Zeit 13](#_Toc405734493)

[2.5.3 Internet Zugriff 13](#_Toc405734494)

[2.5.4 Prüfungsinterne Kommunikation 14](#_Toc405734495)

[2.6 eBook Reader 14](#_Toc405734496)

[2.6.1 Einschränkungen 14](#_Toc405734497)

[3 Analyse 15](#_Toc405734498)

[3.1 Anwendungsfälle 15](#_Toc405734499)

[3.1.1 Übersicht 15](#_Toc405734500)

[3.1.2 Akteure 15](#_Toc405734501)

[3.1.3 AF01 Prüfung erstellen 15](#_Toc405734502)

[3.1.4 AF02 Prüfung absolvieren 15](#_Toc405734503)

[3.1.5 AF02 Abgelegte Prüfung auslesen 15](#_Toc405734504)

[3.2 Nichtfunktionale Anforderungen 15](#_Toc405734505)

[3.2.1 NFA01 Prüfung ist einfach bedienbar 15](#_Toc405734506)

[3.2.2 NFA02 Export erfolgt transparent für den Probanden 15](#_Toc405734507)

[3.2.3 NFA03 Anmeldevorgang dauert weniger als drei Sekunden 15](#_Toc405734508)

[4 Konzept 15](#_Toc405734509)

[4.1 Systemüberblick 15](#_Toc405734510)

[4.1.1 Domänenmodell 16](#_Toc405734511)

[4.2 Export 16](#_Toc405734512)

[4.2.1 Evaluationsmatrix 16](#_Toc405734513)

[4.2.2 Entscheid 16](#_Toc405734514)

[4.3 Kryptographie 17](#_Toc405734515)

[4.3.1 Verschlüsselung 17](#_Toc405734516)

[4.3.2 Passwort hashing 19](#_Toc405734517)

[4.4 Konfigurationsdateien 19](#_Toc405734518)

[4.4.1 Entscheid 19](#_Toc405734519)

[4.5 Import 20](#_Toc405734520)

[4.5.1 Evaluationsmatrix 20](#_Toc405734521)

[4.5.2 Entscheid 20](#_Toc405734522)

[4.6 Digitale Manipulationsmöglichkeiten 21](#_Toc405734523)

[4.6.1 Zeit 21](#_Toc405734524)

[4.6.2 Internet Zugriff 22](#_Toc405734525)

[4.6.3 Prüfungsinterne Kommunikation 22](#_Toc405734526)

[4.7 UI-Design 22](#_Toc405734527)

[4.7.1 Idee 22](#_Toc405734528)

[4.7.2 Login 23](#_Toc405734529)

[4.7.3 Prüfung 23](#_Toc405734530)

[4.8 JavaScript 24](#_Toc405734531)

[4.8.1 SecureExam Library 24](#_Toc405734532)

[4.8.2 CryptoJS Library 24](#_Toc405734533)

[4.8.3 FileSaver.js Library 25](#_Toc405734534)

[4.8.4 Code Obfuscation 25](#_Toc405734535)

[5 Umsetzung 26](#_Toc405734536)

[5.1 Export 26](#_Toc405734537)

[5.2 Kryptographie 26](#_Toc405734538)

[5.2.1 Verschlüsselung 26](#_Toc405734539)

[5.2.2 Entschlüsselung 27](#_Toc405734540)

[5.2.3 Passwort hashing 27](#_Toc405734541)

[5.3 Konfigurationsdateien 28](#_Toc405734542)

[5.3.1 Settings.xml 28](#_Toc405734543)

[5.3.2 SecureExam.xml 28](#_Toc405734544)

[5.4 Import 28](#_Toc405734545)

[5.4.1 XML 28](#_Toc405734546)

[5.4.2 Open Office 28](#_Toc405734547)

[5.5 Digitale Manipulationsmöglichkeiten 28](#_Toc405734548)

[5.5.1 Zeit 28](#_Toc405734549)

[5.5.2 Internet Zugriff 28](#_Toc405734550)

[5.6 UI-Design 29](#_Toc405734551)

[5.6.1 Farben 29](#_Toc405734552)

[5.6.2 Login 29](#_Toc405734553)

[5.6.3 Page-Mode 29](#_Toc405734554)

[5.6.4 Scroll-Mode 30](#_Toc405734555)

[5.7 JavaScript Library 30](#_Toc405734556)

[6 Testing 30](#_Toc405734557)

[6.1 Konzept 30](#_Toc405734558)

[6.2 C# Code 30](#_Toc405734559)

[6.3 HTML / JS Funktionalität 30](#_Toc405734560)

[7 Resultate 30](#_Toc405734561)

[8 Diskussion und Ausblick 30](#_Toc405734562)

[9 Verzeichnisse 31](#_Toc405734563)

[9.1 Literaturverzeichnis 31](#_Toc405734564)

[9.2 Glossar 31](#_Toc405734565)

[9.3 Abbildungsverzeichnis 31](#_Toc405734566)

[9.4 Tabellenverzeichnis 31](#_Toc405734567)

[9.5 Abkürzungsverzeichnis 32](#_Toc405734568)

[10 Anhang 32](#_Toc405734569)

[10.1 Projektmanagement 32](#_Toc405734570)

[10.1.1 Zeitplan 32](#_Toc405734571)

[10.1.2 Sitzungsprotokolle 32](#_Toc405734572)

[10.2 C# Klassendiagramm 32](#_Toc405734573)

[10.3 Bedienungsanleitung 32](#_Toc405734574)

[10.3.1 Konsolenapplikation 32](#_Toc405734575)

[10.3.2 OpenOffice Import 32](#_Toc405734576)

[10.3.3 Generierte Prüfung 32](#_Toc405734577)

[10.4 Weiteres 32](#_Toc405734578)

[10.4.1 CD mit dem vollständigen Bericht als pdf-File und dem SourceCode 32](#_Toc405734579)

# Einleitung

Momentan werden wichtige Prüfungen fast ausschliesslich in Papierform durchgeführt. Dies trotz den immer besser werdenden technischen Möglichkeiten, Prüfungen in digitaler Form durchzuführen zu können.

Nicht nur wäre eine Umstellung auf ein digitales Format eine grosse Erleichterung für die Entlastung der Korrekturarbeit der Professoren, sondern auch für die Studenten. Zum Beispiel könnten Programmierprüfungen in gewohntem Umfeld (über die Tastatur geschriebener Sourcecode) abgehalten werden, oder auch Antworten könnten endlich sauber überarbeitet werden. Diverse dieser Verbesserungsmöglichkeiten sind sehr einfach mit digitalen Prüfungen umsetzbar, doch leider gibt es auch ein paar Schwachpunkte.

Dadurch, dass man digital arbeitet, müssen auch die Manipulationsschutzmechanismen überarbeitet werden. Mit den vielen, heute verfügbaren, technischen Hilfsmitteln, reicht es schon lange nicht mehr aus, einfach die Prüfungsteilnehmer nach Spickzetteln zu durchsuchen.

## Ausgangslage

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

## Problemstellung

* Nicht sicher => manipulationen
* Prüfungsfile kommt abhanden
* Infrastruktur unzuverlässig
* Kommunikationsfähigkeiten
* …

## Zielsetzung / Aufgabenstellung / Anforderungen

### Muss Funktionen

### Wunsch Funktionen

|  |  |
| --- | --- |
| Suchen einer Erinnerung | Durchsuchen der persönlichen Erinnerungen. Kriterien sind Stichworte, Daten, Status und weitere. |
| Filtern der Erinnerungen | Filtern der persönlichen Erinnerungen. Kriterien sind Stichworte, Daten, Status und weitere. |
| Administrationswerkzeug | Mit diesem Werkzeug lassen sich Benutzerkonten verwalten und Änderungen an der Website vornehmen. |

* Formuliert das Ziel der Arbeit
* (Pflichtenheft, Spezifikation)
* (Spezifiziert die Anforderungen an das Resultat der Arbeit)
* (Übersicht über die Arbeit: stellt die folgenden Teile der Arbeit kurz vor)
* (Angaben zum Zielpublikum: nennt das für die Arbeit vorausgesetzte Wissen)

# Grundlagen

## Exportformat

Die Applikation soll eine Exportfunktion anbieten, mit deren Hilfe es dem Benutzer möglich ist, die generierte Prüfung in einem endgültigen Format zu veröffentlichen.

### 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

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, es wird lediglich ein Browser zum Anzeigen benötigt.

## 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 ver-wendet werden kann.

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

## Konfigurationsdateien

Um Einstellungen für das Programm vorzunehmen, wird üblicherweise eine Konfigurationsdatei verwendet, welche, je nach Anwendung, unterschiedlich aussehen kann.

### 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

Eine Form zur Konfiguration von Initialisierungsparametern kann mithilfe einer XML-Datei umgesetzt werden. Es dient zur hierarchischen Strukturierung von Daten und ist plattformunabhängig.

Das Format dieser Dateien gibt vor, wie die Parameter erfasste werden müssen.

## Import

Unsere SecureExam-Applikation stellt eine Importfunktion bereit, welche es dem Benutzer ermöglicht die Fragen für den Test dem Programm zu liefern. Wir haben für diese Arbeit mehrere solcher Importmöglichkeiten untersucht.

### XML

Das XML-Format kann verwendet werden, um durch eine vorgegebene Struktur die benötigten Daten bereitzustellen und von einem Parser einzulesen.

### 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. Diese XML Elemente basieren auf HTML-Elementen.

### 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.

### HTML

Die Struktur von HTML ermöglicht es diese Dateien einzulesen und ihre Gliederung in eine andere Struktur zu übernehmen. Ähnlich wie bei XML kann eine solche Datei von einem Parser verarbeitet und die benötigten Daten eingelesen werden.

## 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

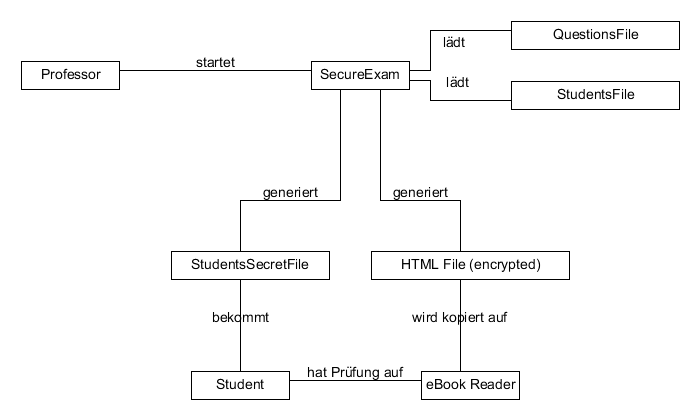
## Systemüberblick



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



**Braucht wohl Überarbeitung… konfig files => export**

## Export

### Evaluationsmatrix



### 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(vgl. 4.2.1), 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

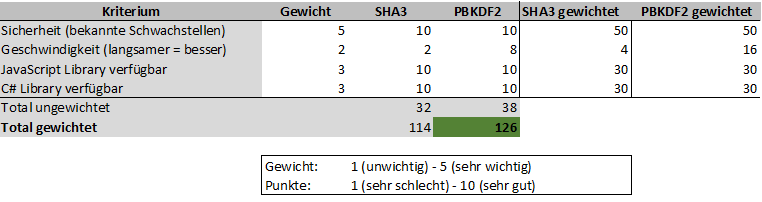


Tabelle : Vergleich SHA3 und PBKDF2

#### Entscheid

Die beiden Algorithmen sind praktisch gleichauf. PBKDF2 ist ein Algorithmus, welcher absichtlich langsam, für das generieren eines Keys aus einem Passwort entwickelt wurde. Trotzdem wird bei SecureExam SHA3 als Hash-Algorithmus eingesetzt. Der Entscheid wurde so getroffen, um eine bessere Anpassung an langsame Hardware (eBook Reader) zu ermöglichen.

#### 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 möglich eine nützliche und lesbare Struktur zu definieren und bietet keine Nachteile zur Verwendung einer INI-Datei. Deshalb wurde der Entscheid gefällt, XML einzusetzen.

## Import

### Evaluationsmatrix

### Entscheid

## 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

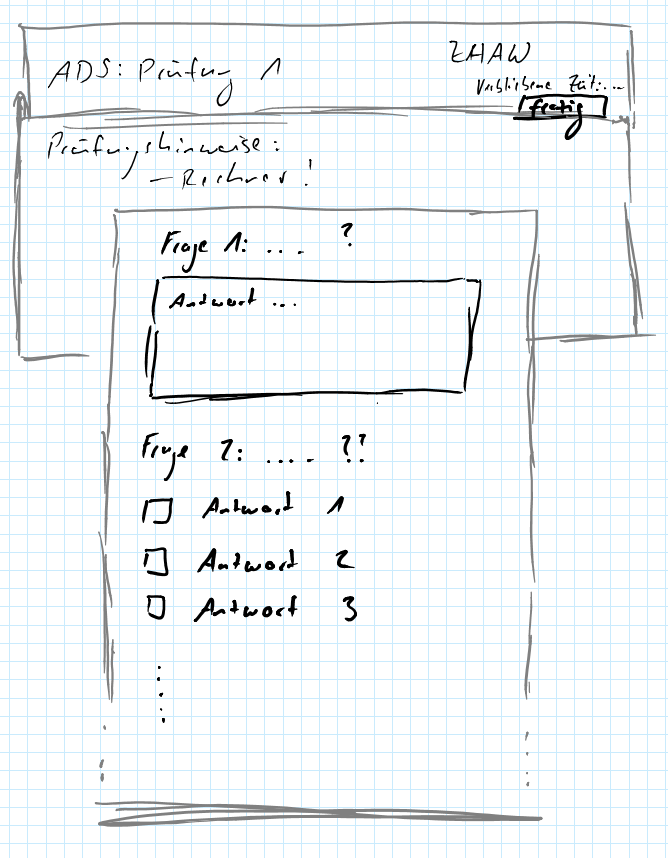
#### Page-Mode



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

blabla

## 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

### Settings.xml

### SecureExam.xml

## Import

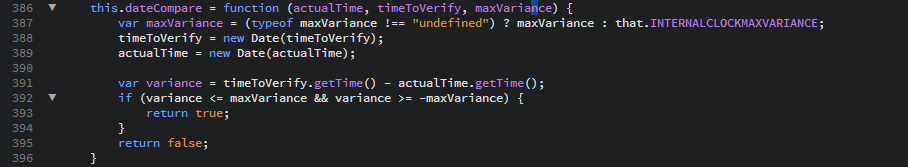
### XML

### Open Office

XSLT über eigenen Parser bla bli bla bla

## Digitale Manipulationsmöglichkeiten

### Zeit



### Internet Zugriff



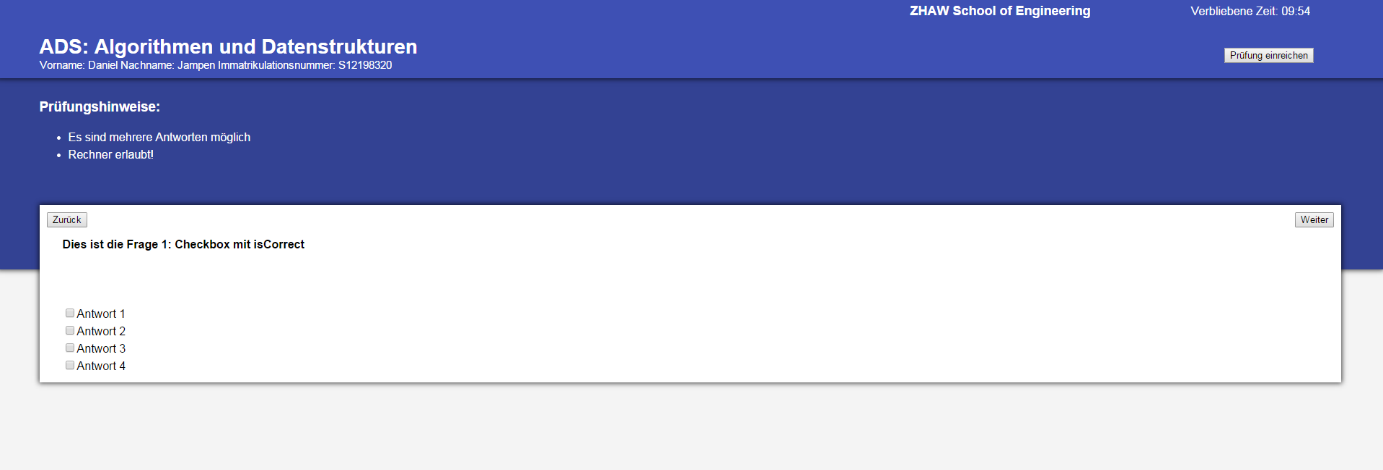
## UI-Design

### Farben

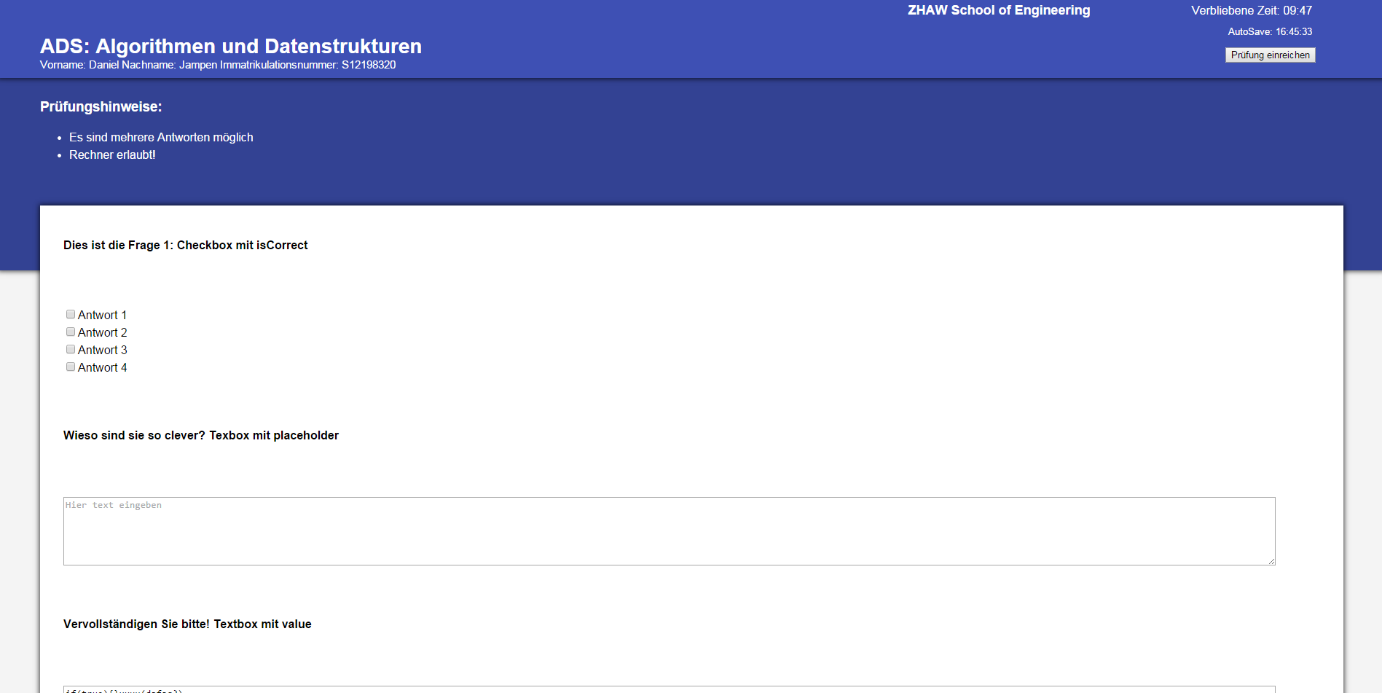
### Login



### Page-Mode



### Scroll-Mode



## JavaScript Library

# Testing

## Konzept

## C# Code

## HTML / JS 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](#_Toc405734282)

[Abbildung 2: ECB Verschlüsselungsmodus 11](#_Toc405734283)

[Abbildung 3: CBC Verschlüsselungsmodus 11](#_Toc405734284)

[Abbildung 4: RSA Verschlüsselung 11](#_Toc405734285)

[Abbildung 5: Systemüberblick 15](#_Toc405734286)

[Abbildung 6: Multi-User AES Verschlüsselung 17](#_Toc405734287)

[Abbildung 7: Multi-User AES Entschlüsselung 18](#_Toc405734288)

[Abbildung 8: Hashing Ablauf 19](#_Toc405734289)

[Abbildung 9: Manipulation der internen Uhr detektieren 21](#_Toc405734290)

[Abbildung 10: Vergleich der internen Uhr mit der Systemuhr 21](#_Toc405734291)

[Abbildung 11: UI Design Idee 22](#_Toc405734292)

[Abbildung 12: Login-Skizze 23](#_Toc405734293)

[Abbildung 13: Skizze Paging Mode 23](#_Toc405734294)

[Abbildung 14: Skizze Scroll Mode 24](#_Toc405734295)

[Abbildung 15: Kryptographie Import in C# 26](#_Toc405734296)

[Abbildung 16: AES mit Using-Scopes 26](#_Toc405734297)

[Abbildung 17: CryptoJS AES decryption mit custom IV & Key 27](#_Toc405734298)

[Abbildung 18: SHA256 Hashing in C# 27](#_Toc405734299)

[Abbildung 19: SHA256 Berechnung mit CryptoJS 28](#_Toc405734300)

## Tabellenverzeichnis

[Tabelle 1: Vergleich AES und RSA 17](#_Toc405734301)

[Tabelle 2: Vergleich SHA3 und PBKDF2 19](#_Toc405734302)

[Tabelle 3: JavaScript Obfuscation 25](#_Toc405734303)

## Abkürzungsverzeichnis

# 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