

# Elektronische Wahlen mit bedingungslosem Wahlgeheimnis

Dokumentation der Erkenntnisse der Bachelor-Thesis

Studiengang: BSc Informatik  
Autoren: Bürk Timo, Nellen Sebastian  
Betreuer: Prof. Dr. Haenni Rolf, Prof. Dr. Locher Philipp  
Experte: Dr. Flueckiger Federico  
Datum: 19.01.2017

## Versionen

Version	Datum	Status	Bemerkungen
0.1	06.01.2017	Entwurf	Vorabversion erstellt
0.2	13.01.2017	Entwurf	Texte eingepflegt
0.3	14.01.2017	Entwurf	Review-Version erstellt
0.4	14.01.2017	Entwurf	Review-Befunde eingepflegt
0.5	15.01.2017	Entwurf	2. Review-Version erstellt
0.6	17.01.2017	Entwurf	Review-Befunde eingepflegt
0.7	18.01.2017	Entwurf	Rohfassung erstellt, letzte Anpassungen gemacht
1.0	19.01.2017	Definitiv	Abgabeversion erstellt

# Management Summary

Die vorliegende Dokumentation der Bachelor-Thesis beschreibt die Planung und Realisierung eines Prototypen zur Erstellung elektronischer Wahlen. Als theoretische Grundlage dient dabei das E-Voting Protokoll, welches in der wissenschaftlichen Arbeit von Haenni und Locher „*Verifiable Internet Elections with Everlasting Privacy and Minimal Trust*“ beschrieben wird. Das Protokoll bietet unter anderem die Eigenschaften des bedingungslosen Wahlgeheimnisses und beinhaltet schwächere Vertrauensannahmen gegenüber Dritten.

Bisherige Implementierungen von elektronischen Wahlsystemen benötigen vertrauenswürdige Drittparteien. Diese Drittparteien haben einen grossen Einfluss auf die Wahl und können diese sogar manipulieren. Es geht in dieser Arbeit allerdings nicht um Verschwörungstheorien sondern um den Fakt, dass offene Strukturen inhärent sicherer und nachvollziehbarer sind als verschleierte. Durch das offene Design des Protokolls und damit des implementierten Prototyps wird die Nachvollziehbarkeit von elektronischen Wahlen auf eine neue Ebene erhoben. Dazu kommt der Fakt, dass durch die Verwendung von kryptographischen Elementen, welche nicht abhängig von Rechenleistung sind, ein fortwährendes, bedingungsloses Wahlgeheimnis aufgebaut wird.

Diese Diplomarbeit zeigt anhand von Use Cases wie das Protokoll in der Praxis realisiert werden kann. Das Protokoll wurde mit Hilfe der BFH-eigenen Unicrypt Software-Library realisiert und in vier Applikationen aufgeteilt. Durch den modularen Aufbau des Prototypen können Komponenten unabhängig weiterentwickelt oder sogar ausgetauscht werden. Als zentrale Komponente dient das Bulletin Board, eine Art digitales öffentliches Anschlagbrett, welches Informationen für alle beteiligten Parteien annimmt und zur Verfügung stellt. Die Wahladministration definiert mit Hilfe der AdminApp die Wahlen und legt fest, wer stimmberechtigt ist. Des Weiteren gibt es eine VoterApp, mit welcher der Wähler sich registrieren und an Wahlen teilnehmen kann. Schliesslich haben wir eine VerifierApp implementiert, welche das Stimmmaterial verifiziert und das Resultat publiziert.

Damit die Wahlerstellung, die Wählerregistrierung und die Publikation des Resultates gegen Manipulationen geschützt sind, werden die Nachrichten digital signiert und durch einen verschlüsselten Kanal gesendet. So kann sichergestellt werden, dass einerseits nur Nachrichten von registrierten Personen vom Bulletin-Board angenommen werden und andererseits, dass die Datenpakete nicht von Dritten auf dem Transportweg eingesehen oder verändert werden können. Das Protokoll bietet bereits eine Anonymisierung basierend auf kryptographischen Elementen. Trotzdem wird ein alterner Kanal gewünscht, weil Versand und Empfang von Datenpaketen mögliche Rückschlüsse auf deren Herkunft erlauben. Deshalb wurde zur erweiterten Sicherstellung der Wähleranonymität schliesslich die Funktionalität implementiert, Stimmen anonym über das Anonymisierungsnetzwerk Tor abzugeben.

Dieses Dokument gibt einen Überblick über die geleistete Arbeit im Rahmen der Bachelor-Thesis. Neben einer Einleitung zum Thema enthält es eine Aufstellung der Ziele und des Zeitplans. Weiter werden die theoretischen Aspekte erläutert und Begrifflichkeiten eingeführt. Danach folgt eine High-Level, sowie eine detaillierte Produktbeschreibung. Das Dokument schliesst mit einem Ausblick und einem Fazit.



# Inhaltsverzeichnis

<b>Management Summary</b>	i
<b>1. Einleitung</b>	1
<b>2. Projektmanagement</b>	3
2.1. Ziele . . . . .	3
2.2. Abgrenzungen . . . . .	4
2.3. Zeitplan . . . . .	5
<b>3. Theoretische Aspekte</b>	7
3.1. Begrifflichkeiten . . . . .	7
3.2. E-Voting Protokoll nach Haenni und Locher . . . . .	7
3.3. Tor als möglicher anonymer Kanal . . . . .	10
<b>4. Produktbeschreibung</b>	13
4.1. Use Cases . . . . .	13
4.2. High-Level Aufbau . . . . .	14
4.3. Technologiewahl . . . . .	14
<b>5. Detaillierter Aufbau</b>	17
5.1. Bulletin-Board . . . . .	17
5.2. Util-Package . . . . .	19
5.3. Architektur der Java-Applikationen . . . . .	22
5.4. AdminApp . . . . .	23
5.5. VoterApp . . . . .	23
5.6. VerifierApp . . . . .	24
5.7. Testing . . . . .	25
5.8. Integration von Tor . . . . .	25
<b>6. Ausblick</b>	29
<b>7. Fazit</b>	31
<b>Selbständigkeitserklärung</b>	33
<b>Literaturverzeichnis</b>	35
<b>Abbildungsverzeichnis</b>	37
<b>A. Bachelorthesis-Aufgabe</b>	39
<b>B. Use Cases</b>	41
<b>C. Anforderungen an Komponenten</b>	51

<b>D. Setup Guide</b>	<b>53</b>
<b>E. JSON-Beispieldateien</b>	<b>59</b>
<b>F. Test Cases</b>	<b>65</b>
<b>G. Sitzungsprotokolle</b>	<b>71</b>

# 1. Einleitung

Das Bedürfnis nach elektronischen Wahlen besteht schon seit längerem. Oft werden elektronische Wahlen auch als Lösung für die Mobilisierung der jüngeren Wählerschaft genannt. Zudem setzt sich der Trend zur Digitalisierung in allen Lebenslagen weiter fort. Es liegt auf der Hand, dass sich damit auch elektronische Wahlen verbreiten werden, was den Bedarf an kryptographisch sicheren Wahlsystemen weiter erhöht. Trotz den veränderten Bedürfnissen müssen demokratische Eckpfeiler wie das Wahlgeheimnis oder die Sicherstellung der Wahlberechtigung nach wie vor gewährleistet sein. An diesem Punkt setzt unsere Bachelor-Thesis an, welche einen Proof of Concept eines Wahlsystems mit bedingungslosem Wahlgeheimnis zum Ziel hat.

**Ausgangslage** Bei den aktuellen Implementierungen von elektronischen Wahlsystemen werden vertrauenswürdige Drittparteien (trusted parties) benötigt. Diese verwalten beispielsweise die geheimen Schlüssel einer Wahl und haben somit Einsicht in die Wahldaten. Im Gegensatz zur heutigen Papierwahl werden die Stimmen nicht von unabhängigen Wahlhelfern ausgezählt, sondern von einer Software. Da der Programmcode und das Infrastruktur-Design nicht öffentlich einsehbar sind, muss den Entwicklern und Betreibern ein hohes Vertrauen entgegengebracht werden. Zusätzlich werden in den heutigen Wahlsystemen Verschlüsselungstechniken genutzt, welche auf heute nicht lösbareren Problemen beruhen. Es kann daher nicht sichergestellt werden, dass die Wahldaten auch in Zukunft sicher sind.

Haenni und Locher haben in ihrer wissenschaftlichen Arbeit „*Verifiable Internet Elections with Everlasting Privacy and Minimal Trust*“ [2] beschrieben, wie die Vertrauensannahmen in die Drittparteien vermindert werden können und gleichzeitig ein bedingungsloses Wahlgeheimnis sichergestellt werden kann. Gemäss dem Protokoll, registriert sich der Wähler beim Bulletin-Board (eine Art digitales Anschlagbrett) und publiziert dort sein *Public Voter Credential*. Die Wahladministration kann daraufhin eine neue Wahl erstellen und die berechtigten Wähler zu einer Wahl zulassen. Anschliessend kann ein Wähler seine Stimme zu einer Wahl abgeben. Am Ende einer Wahl kann jeder und jede selbständig die Wahl überprüfen und auszählen, wobei diese Überprüfung ein nicht unerheblicher Rechenaufwand bedeutet.

**Beitrag dieser Arbeit** Im Rahmen der Arbeit wurde ein Proof of Concept des Protokolls von Haenni und Locher implementiert. Neben der schwächeren Vertrauensannahme bietet das Protokoll ein Stimmgeheimnis, welches auch in Zukunft nicht gebrochen werden kann. Das Ziel dieser Arbeit war die Entwicklung eines Grundsystems, welches die verschiedenen Komponenten zur Durchführung einer sicheren elektronischen Wahl umfasst.

Der Prototyp wurde modular konzipiert und deshalb in vier Applikationen aufgeteilt. Als zentrale Komponente dient das Bulletin-Board, welches Informationen für alle beteiligten Parteien annimmt und zur Verfügung stellt. Die Wahladministration definiert mit Hilfe der *AdminApp* die Wahlen und legt fest, wer stimmberechtigt ist. Des Weiteren gibt es eine *VoterApp*, mit welcher der Wähler sich registrieren und an Wahlen teilnehmen kann. Schliesslich wurde eine *VerifierApp* implementiert, welche das Stimmmaterial verifiziert und das Resultat publiziert. Das Bulletin-Board wurde in PHP implementiert und basiert auf dem Slim-Framework. Die anderen drei Applikationen wurden in Java implementiert und basieren auf der BFH-eigenen *Unicrypt* Software-Library auf.

Damit die Wahlerstellung, die Wählerregistrierung und die Publikation des Resultates gegen Manipulationen geschützt sind, werden die Nachrichten digital signiert und durch einen verschlüsselten Kanal gesendet. So kann sicher gestellt werden, dass einerseits nur Nachrichten von registrierten Personen vom Bulletin-Board angenommen werden und andererseits, dass die Datenpakete nicht von Dritten auf dem Transportweg eingesehen oder verändert werden können. Zur erweiterten Sicherstellung der Wähleranonymität wurde die Funktionalität implementiert, Stimmen über das Anonymisierungsnetzwerk Tor abzugeben.

**Inhalt des Dokuments** Dieses Dokument zeigt im Kapitel 2 die Projektmanagement-Aspekte dieser Thesis. Es werden zuerst die verschiedenen Ziele dieser Arbeit beschrieben und danach die Planung mit Projekt-Milestones aufgezeigt. Im folgenden Kapitel 3 werden die theoretischen Aspekte rund um die Arbeit aufgezeigt. Neben einigen Begrifflichkeiten, werden auch der Protokoll-Ablauf und die theoretischen Aspekte rund um das Anonymisierungsnetzwerk Tor erörtert, welches zur anonymen Stimmabgabe verwendet wird.

Das nachfolgende Kapitel 4 gibt einen Überblick über den implementierten Prototypen namens „abcVote“. Zuerst werden die definierten Use Cases beschrieben, danach wird der High-Level Aufbau und die verwendeten Technologien aufgezeigt. Der detaillierte Aufbau des Proof of Concept und die Implementation von Tor folgt im darauf folgenden Kapitel 5.

Dieses Dokument bietet zudem im Kapitel 6 einen Ausblick über mögliche Erweiterungen und Anforderungen, welche für ein produktives Wahlsystem nötig sind. Schliesslich folgt ein Fazit zu dieser Bachelor-Thesis im Kapitel 7.

## 2. Projektmanagement

Das folgende Kapitel beschreibt die Projektmanagement-Aspekte dieser Bachelor-Thesis. Neben den definierten Zielen werden die planerischen Eckdaten, wie zum Beispiel die Milestones und die Zeitplanung, aufgezeigt.

### 2.1. Ziele

Die Ziele wurden in zwei Kategorien aufgeteilt. Zum Einen wurden „Muss-Ziele“ definiert, welche als Pflichtteil dieser Arbeit zu verstehen sind, und zum Anderen „Kann-Ziele“, welche bei noch vorhandener Zeit ebenfalls adressiert werden können.

**Teil 1: Wahl durchführen (Muss)** Das Hauptziel dieser Arbeit ist ein Proof of Concept eines Wahlsystems gemäss dem Protokoll von Haenni und Locher [2]. Das System sollte modular aufgebaut werden und besteht aus vier Komponenten. Die erste Komponente enthält die Funktionen für die Wahladministration und muss somit die Möglichkeit bieten eine Wahl zu erstellen und die berechtigten Wähler auswählen zu können. Der Wähler erhält eine eigene Komponente, mit welcher er sich registrieren und an einer Abstimmung teilnehmen kann. Der Verifier verifiziert und zählt eine Wahl aus. Er erhält ebenfalls eine eigenständige Komponente, mit welcher er Wahldaten verifizieren und auszählen kann. Schliesslich muss mit dem Bulletin-Board eine zentrale Komponente zur Annahme sowie Publikation der Wahlen, Wahldaten und Resultaten implementiert werden. Die untenstehende Abbildung 2.1 bietet eine Übersicht über die Anforderungen an den Proof of Concept, welche zu Projektstart diskutiert wurden.

Teil 1: Wahl durchführen		Muss
§ Komponente 1:	Admin-App zur Wahl-Definition	
	<ul style="list-style-type: none"><li>• Wahlverfahren: z.B. 1 aus 2, 1 aus n, k aus n</li><li>• n muss festgelegt werden können</li><li>• Wahl organisieren (Kandidaten definieren)</li><li>• Wählerliste eingeben</li></ul>	
§ Komponente 2:	Client für Stimmabgabe und Registrierung (Java, Unicrypt)	
	<ul style="list-style-type: none"><li>• Wählerauthentisierung: Signierung der Stimme mit PublicKey einer vorhandenen PKI</li><li>• Wahldaten beziehen von Bulletin-Board und darstellen</li><li>• Verwaltung der Credentials (Private and Public Voter Credential)</li><li>• Stimmen generieren und versenden</li></ul>	
§ Komponente 3:	Bulletin-Board (basierend auf MySQL-REST-Lösung)	
	<ul style="list-style-type: none"><li>• Annahme und Ablage der Stimmen</li><li>• Wahlperiode (Konsequenz von zu früh oder zu spät!)</li><li>• Umgang mit Mehrfach-Stimmen: Entweder erster oder letzter Ballot oder keine zählt</li><li>• Wahl-Identifier (unique), für verschiedene Wahlen</li><li>• Zentrales System, nur Bulletin Board online! Sämtliche Kommunikation läuft über Bulletin-Board</li></ul>	
§ Komponente 4:	Wahl auszählen und verifizieren (Java, Unicrypt)	
	<ul style="list-style-type: none"><li>• Unabhängige Kompetenz, welche sowohl bei Client als auch Admin benutzt werden kann</li><li>• Umgang mit Mehrfach-Stimmen: Entweder erster oder letzter Ballot oder keine zählt</li></ul>	
	<ul style="list-style-type: none"><li>- Kommunikationsprotokoll zwischen Komponenten muss spezifiziert werden (z.B. via JSON, XML, etc.)</li><li>- Publikation der Wahldaten vom Admin -&gt; Bulletin Board, Client bezieht Daten ebenfalls vom Bulletin Board</li><li>- Wähler müssen sich vor Wahl-Definition melden. Bei der Wahl-Definition kann der Admin aus vorhandenen Credentials wählen.</li></ul>	

Abbildung 2.1.: Screenshot der Ziele des Teil 1, Durchführen einer Wahl

**Teil 2a: Recherche anonymer Kanal (Muss)** Das Sekundärziel dieser Arbeit ist die Recherche be treffend der Stimmabgabe über einen anonymen Kanal. Auch wenn die Stimmabgabe über einen verschlüsselten Kanal abläuft und die Identität des Wählers durch das Protokoll geschützt ist, kann nicht ausgeschlossen werden, dass jemand im Netzwerk mithört respektive mitliest. Beispielsweise kann ein Angreifer feststellen, dass ein Wähler gerade etwas an das Bulletin-Board gesendet hat. Zudem kann der Angreifer aufgrund der öffentlichen Zugänglichkeit des Bulletin-Boards erkennen, wenn eine Stimme eintrifft. Mit diesen beiden Informationen kann ein Angreifer den Wähler mit seiner Stimme in Verbindung bringen.

**Teil 2b: Implementation anonymer Kanal (Kann)** Sofern die Recherche betr. eines anonymen Kanals positiv verlaufen ist, soll dieser in den Proof of Concept integriert werden.

**Teil 3a: „Append-only“ Bulletin-Board (Kann)** Severin Hauser, Mitglied der E-Voting-Gruppe der BFH, hat ein Bulletin-Board entwickelt, welches die „Append-only“-Eigenschaft besitzt. Dies bedeutet, dass keine Daten vom Bulletin-Board mutiert oder gelöscht werden können, sondern nur angefügt (engl. to append) werden können. Das im Rahmen dieses Projektes entwickelte Bulletin-Board besitzt diese Eigenschaft nicht, was ein erhöhtes Vertrauen in den Infrastruktur-Betreiber zur Folge hat. Es wäre wünschenswert, wenn dieses „Append-only“-Bulletin-Board in den Prototypen integriert werden könnte.

**Teil 3b: Verschlüsselung der Stimmen (Kann)** Der Grundausbau des Wahlsystems bietet lediglich die Möglichkeit Stimmen signiert und über einen verschlüsselten Kanal an das Bulletin-Board zu übertragen. Ein Angreifer kann somit die aktuell übertragene Stimme nicht lesen oder manipulieren. Da die Stimmen aber unverschlüsselt abgespeichert werden, ist die Fairness der Wahl nicht gegeben. Dies bedeutet, dass bei einer noch laufenden Wahl das aktuelle Wahlresultat berechnet werden kann. Um dies zu verhindern, soll die Möglichkeit geboten werden, die Stimmen zu verschlüsseln und erst am Wahlende zu entschlüsseln.

## 2.2. Abgrenzungen

Im folgenden Abschnitt wird definiert, welche Aspekte Teil dieser Bachelor-Thesis sind, und welche als gegeben vorausgesetzt werden. Dazu wurden zum einen die Anforderungen und Voraussetzungen des E-Voting Protokolls von Haenni und Locher betrachtet und zum anderen die Rahmenbedingungen des Projekts miteinbezogen. Folgende Abgrenzungen wurden definiert:

- Die Geräte, welche für die Stimmabgabe verwendet werden, werden als sicher vorausgesetzt. Das Schützen dieser Geräte ist somit nicht Teil des Projekts.
- Für den Registrierungsprozess wird ein authentischer Kanal zwischen Wähler und Bulletin-Board verlangt. Dieser wird umgesetzt, indem die Registrierung vom Wähler signiert wird. Dazu muss auf dem Bulletin-Board eine PKI vorhanden sein. Diese wird im Rahmen des Projekts als vorhanden angenommen. Dies bedeutet, dass manuell eine Liste mit Zertifikaten als PKI erstellt wird und das Wahlsystem keine Funktionen für die Verwaltung dieser PKI bietet.
- Fokus des Projekts ist in erster Linie die Umsetzung des E-Voting Protokolls. Daher werden nur einfache Wahlen umgesetzt bei denen aus  $n$  Möglichkeiten  $k$  ausgewählt werden können. Weitere Wahltypen sowie das Einbinden von Zusatzinformationen werden nicht abgebildet.

## 2.3. Zeitplan

Zu Beginn der 16-wöchigen Projektarbeit wurden die Milestones dieser Thesis festgelegt. Die Milestones wurden in der Regel im Abstand von zwei Wochen festgelegt, da die Treffen zwischen Studenten und Betreuern jeweils alle zwei Wochen geplant waren. Im Anhang G sind die Protokolle der Sitzungen zwischen Betreuer und Studenten abgedruckt.

Woche	Milestone
1	Kick-Off Meeting mit Betreuern
5	Ready to Implement
7	Eine Wahl kann erstellt werden.
9	Eine Stimmabgabe kann durchgeführt werden.
11	Ein Wahlresultat kann verifiziert und berechnet werden.
14	Abgabe Dokumentations-Entwurf an Betreuer
16	Abgabe Dokumentation und Kurzpräsentation

Tabelle 2.1.: Milestones

Angelehnt an die Milestones (Tabelle 2.1) wurde folgender Soll-Zeitplan definiert:

Woche	Tätigkeiten
1	Vorbereiten Kick-Off Meeting, Planung Grobkonzept & zeitlicher Ablauf
2-3	Detail-Spezifikation des Produkts (Schnittstellen & Komponenten), Use Cases erstellen, Technologie-Wahl & Wahl der Entwicklungsumgebung (IDE), Tests definieren
4-5	Umgebung aufsetzen (IDE, Server, Infrastruktur, ...), Schnittstellen und APIs aufbauen, Definition Kommunikations-Layer
6-7	Umsetzung Teil 1: Wahl durchführen (AdminApp & Teil Bulletin-Board)
8-9	Umsetzung Teil 1: Wahl durchführen (VoterApp & Teil Bulletin-Board)
10-11	Umsetzung Teil 1: Wahl durchführen (VerifierApp & Teil Bulletin-Board), Testing Teil 1
12-13	Recherche Teil 2a: anonymer Kanal, Reserve Umsetzung Teil 1
14-15	Dokumentation, Umsetzung Teil 2b: anonymer Kanal (falls Recherche positiv), Recherche Teil 3a: Append-only Bulletin-Board, Recherche Teil 3b: Verschlüsselung der Votes
16	Dokumentation, Vorbereiten der Präsentationen

Tabelle 2.2.: Soll-Zeitplan

Der Soll-Zeitplan (Tabelle 2.2) konnte grösstenteils eingehalten werden und musste nur punktuell adaptiert werden. Dies ist nicht zuletzt dem Umstand zu verdanken, dass diese Arbeit zweit realisiert wurde.

- Die benötigte Einarbeitungszeit in die Teils neuen Technologien war länger als bei der Planung angenommen. Beispielsweise wurde einige Einarbeitungszeit benötigt bis die Verwendung von JavaFX und dem Slim PHP-Framework im Sinne des Projekt verwendet werden konnte.
- Der Datenaustausch zwischen den Java-Apps und dem Bulletin-Board war nicht immer leicht. Obwohl beide Umgebungen das JSON-Format unterstützen, gab es gerade bei der Übersetzung Probleme.

## 3. Theoretische Aspekte

Im folgenden Kapitel werden zuerst einige Begrifflichkeiten erläutert, um damit das E-Voting Protokoll nach Haenni und Locher [2] detailliert aufzuzeigen. Neben dem Ablauf werden die beiden wichtigen Eigenschaften Everlasting Privacy und Minimal Trust erklärt. Abschliessend gibt das Kapitel einen theoretischen Überblick über Tor als möglicher anonymer Kanal.

### 3.1. Begrifflichkeiten

Nachfolgend werden kurz einige im Protokoll (und somit auch im Proof of Concept) verwendete Begrifflichkeiten erläutert. Im Bericht des Vorprojekts [1] sind die Begriffe detaillierter erklärt.

**Commitment** Bei einem Commitment geht es darum, sich auf einen ausgewählten Wert festzulegen und diesen danach für eine gewisse Zeit zu verstecken. Ein weiterer wichtiger Anspruch an das Commitment ist, dass der ausgewählte Wert nicht verändert werden kann, solange er versteckt ist. Im verwendeten Protokoll werden Pedersen-Commitments verwendet, die „perfectly hiding“ und „computationally binding“ sind. Erstes bedeutet, dass auch mit unbegrenzten Mitteln und Rechenleistung nicht auf das Geheimnis geschlossen werden kann. Zweites bedeutet, dass erst mit sehr hoher beziehungsweise zukünftiger Rechenleistung ein anderer Wert berechnet werden kann, welcher dasselbe Commitment ergibt.

**(Non-Interactive) Zero-Knowledge Proof, (NI)ZKP** Ziel eines Zero-Knowledge Proofs ist es jemandem zu beweisen, dass man ein bestimmtes Geheimnis kennt, ohne dabei das Geheimnis selbst preis zu geben. Die Unterscheidung zwischen „Non-Interactive“ Zero-Knowledge Proof (NIZKP) und „Interactive“ Zero-Knowledge Proof (ZKP) ist hier wichtig, da abgegebene Stimmen nicht *interaktiv* im klassischen Sinne verwendet werden. Interaktiv wäre, wenn der Wähler direkt mit einer Wahladministration kommuniziert und den Beweis nur für diese überprüfbar sein muss. Dies ist im Protokoll von Haenni und Locher nicht gegeben, da ein Wähler den Zero-Knowledge Proof of Knowledge berechnet, anschliessend an das Bulletin-Board sendet und erst im Nachhinein von irgendjemandem überprüft wird.

**Fairness** Eine Wahl ist fair, wenn jeder Wähler seine Stimme unabhängig abgeben kann ohne zu wissen, wie das aktuelle Wahl-Resultat lautet. In einer klassischen Wahl gibt es zwar Trendrechnungen, welche aber nur Trends angeben und keine verlässliche Aussage über den aktuellen Stand abgeben. Bei einer elektronischen Wahl wird eine vertrauenswürdige Drittpartei benötigt, welche das Resultat bis zum Ende der Wahl geheim hält.

### 3.2. E-Voting Protokoll nach Haenni und Locher

Die nachfolgende Beschreibung des Protokollablaufs basiert auf den im Vorprojekt [1] von den Autoren erarbeiteten Erkenntnissen.

### 3.2.1. Ablauf Protokoll

Das Protokoll kann grob in vier Phasen gegliedert werden.

**Registrierung** Während der Registrierungsphase generiert jeder *Wähler*  $V$  folgende Größen:

- *Private Voter Credential*: Der geheime Teil des Voter Credential besteht aus den beiden Werten  $\alpha$  und  $\beta$ , welche zufällig aus der Restklassenmenge  $\mathbb{Z}$  mit Basis  $q$  gewählt werden.  
 $(\alpha, \beta) \in_r \mathbb{Z}_q^2$
- *Public Voter Credential*  $u$ : Der öffentliche Teil des Voter Credential  $u$  wird aus den beiden Werten  $\alpha$  und  $\beta$  und den beiden Generatoren  $h_1$  und  $h_2$  berechnet. Letztere sind öffentlich bekannte Werte und Teil der aktuellen Abstimmung.  
 $u = h_1^\alpha h_2^\beta \in G_q$

Nachdem der Wähler diese beiden Teile seines Voter Credentials generiert hat, sendet er den öffentlichen Teil  $u$  über einen authentischen Kanal an die Wahladministration.

**Abstimmungsvorbereitung** Die Wahladministration definiert und publiziert die Liste aller zu der aktuellen Wahl zugelassenen Wähler  $U = ((V_1, u_1), \dots, (V_M, u_M))$

Zudem berechnet sie den Koeffizienten  $A = (a_1, \dots, a_M)$  mit dem Polynom  $P(X) = \prod_{i=1}^M (X - u_i) \in \mathbb{Z}_p[X]$ . Der Koeffizient  $A$  wird von den Wählern benötigt, um zu beweisen, dass sie zu dieser Wahl berechtigt sind (ein so genannter *Membership Proof*). Mit Hilfe der Wählerliste  $U$  könnte theoretisch jeder den Koeffizienten  $A$  nachrechnen. Da diese Berechnung rechenintensiv ist, wird sie allerdings nur einmalig von der Wahladministration berechnet.

Abschliessend wählt die Wahladministration einen unabhängigen Wahlgenerator  $\hat{h} \in G_q$  und publiziert diesen gemeinsam mit  $U$  und  $A$  als Tripel  $(U, A, \hat{h})$  auf dem öffentlichen Bulletin-Board.

**Stimmabgabe** Der Wähler  $V$  trifft seine Entscheidung - er stimmt ab - und erzeugt seine Stimme  $e$ . Danach wird die Stimme encodiert. Das Protokoll schreibt nicht vor, wie die Stimme  $e$  encodiert sein muss. Der Wähler  $V$  berechnet nun verschiedene Werte, welche nachfolgend beschrieben werden.

- Die Election Credential  $\hat{u} = \hat{h}^\beta \in G_q$
- Commitment zum Public Voter Credential  $c = \text{com}_p(u, r)$  mit  $r \in_R \mathbb{Z}_p$
- Commitment zum Private Voter Credential  $d = \text{com}_q(\alpha, \beta, s)$  mit  $s \in_R \mathbb{Z}_q$
- Set membership proof  $\pi_1$   
 $\pi_1 = \text{NIZK}P_e[(u, r) : c = \text{com}_p(u, r) \wedge P(u) = 0]$   
 $\pi_1$  beweist, dass das Commitment zum Public Voter Credential  $c$  zu einem Public Voter Credential  $u$  aus der Wahlliste  $U$  gehört.
- Proof of known representation of a committed value  $\pi_2$   
 $\pi_2 = \text{NIZK}P_e[(u, r, \alpha, \beta, s) : c = \text{com}_p(u, r) \wedge d = \text{com}_q(\alpha, \beta, s) \wedge u = h_1^\alpha h_2^\beta]$   
 $\pi_2$  beweist, dass derselbe Wähler die beiden Commitments  $c$  und  $d$  berechnet hat.
- General Preimage Equality Proof  $\pi_3$   
 $\pi_3 = \text{NIZK}P_e[((\alpha, \beta, s) : d = \text{com}_q(\alpha, \beta, s) \wedge \hat{u} = \hat{h}^\beta)]$   
 $\pi_3$  zeigt, dass dasselbe  $\beta$  zur Berechnung des Commitments zum Private Voter Credential  $d$  und zur Berechnung des Election Credentials  $\hat{u}$  verwendet wurde. Zudem wird mit  $\pi_3$  gezeigt, dass die Stimme  $e$  zu dieser Wahl gehört, und dass es sich nicht um ein Duplikat handelt.

Die obengenannten Beweise  $\pi_1$ ,  $\pi_2$  und  $\pi_3$  werden jeweils in Verbindung mit der Stimme  $e$  gebracht, weshalb  $e$  jeweils bei den Non-Interactive Zero-Knowledge Proofs *NIZKP* tiefgestellt hingeschrieben ist. Diese Verknüpfungen bewirken, dass die Stimme  $e$  nicht unbemerkt verändert werden kann.

Abschliessend schickt der Wähler den Stimmzettel (Ballot)  $B$  über einen anonymen Kanal zum Bulletin-Board. Der elektronische Stimmzettel  $B$  enthält somit die Commitments zum Public und Private Voter Credential  $c$  und  $d$ , die Stimme  $e$  und das Election Credential  $\hat{u}$ , als auch die drei Beweise (proofs)  $\pi_1$ ,  $\pi_2$  und  $\pi_3$ .

**Öffentliche Auszählung** Am Ende der Wahl müssen sämtliche eingegangenen digitalen Stimmzettel (Ballots) überprüft werden, analog einer traditionellen Wahl in der Schweiz. Jeder und jede kann die Überprüfung als auch die Auszählung selbst durchführen und so verifizieren. Einerseits müssen ungültige Ballots als solche gekennzeichnet werden. Sie dürfen nicht gelöscht werden, damit jeder selbständig nochmals die Korrektheit resp. die Ungültigkeit aller Ballots überprüfen kann. Andererseits müssen doppelte Stimmabgaben über das Election Credential  $\hat{u}$  identifiziert werden. Sofern nicht alle Stimmabgaben identisch sind, werden Konflikte anhand vordefinierter Prozesse gelöst.

### 3.2.2. Wichtige Eigenschaften

**Everlasting Privacy** Wenn man von Everlasting Privacy im Zusammenhang mit dem Protokoll redet, müssen zwei mögliche Arten von Gegnern unterschieden werden.

Ein *heutiger Gegner* handelt vor oder während einer Abstimmung. Seine Ziele sind auf der einen Seite das Brechen der Integrität, in dem der Gegner eine valide Stimme im Namen eines Anderen abgibt. Andererseits versucht der Gegner die Geheimhaltung einer Wahl zu brechen, in dem er versucht, Stimmen mit Wählern in Verbindung zu bringen.

Er hat nur eine beschränkte Rechenleistung und ist *polynomially bounded*<sup>1</sup> und kann weder diskrete Logarithmen noch aktuelle kryptographische Hash-Funktionen brechen. Somit kann er kein  $\alpha'$  und  $\beta'$  berechnen, welches dasselbe Public Voter Credential  $u$  ergeben würde. Der Gegner kann zudem keine valide Stimme erzeugen, solange er kein solches Paar  $\alpha'$  und  $\beta'$  kennt.

Zugleich kann ein heutiger Gegner mit seinen Mitteln nur eine Replay-Attacke ausführen. Diese Attacke lässt sich allerdings einfach feststellen, da zwei identische Stimmzettel mit demselben Election Credential  $\hat{u} = \hat{h}^\beta$  eingegangen sind. Abschliessend kann festgehalten werden, dass ein heutiger Gegner keine Bedrohung für die *everlasting privacy* darstellt.

Ein *zukünftiger Gegner* wird irgendwann in der Zukunft nach einer Wahl aktiv werden und versuchen eine Stimme mit einem Wähler in Verbindung zu bringen. Im Gegensatz zum heutigen Gegner ist er an keine Rechenleistung gebunden, da man heute nicht mit Gewissheit sagen kann, welche Technologien in der Zukunft möglich sein werden.

Der zukünftige Gegner stellt aber keine Gefahr für die *everlasting privacy* dar. Auch wenn dieser Gegner diskrete Logarithmen effizient berechnen und damit  $\beta$  aus  $\hat{u} = \hat{h}^\beta$  berechnen kann, kann trotzdem nicht eindeutig auf den Wähler geschlossen werden. Denn  $u$  in  $u = h_1^\alpha h_2^\beta$  besitzt jeweils eine Lösung für jedes Paar  $(u_i, \beta_i)$ . Dazu kommt, dass alle Beweise (Proofs) in den Stimmzetteln (Ballots) *perfectly hiding* sind. Es besteht aber trotzdem ein Risiko, dass ein Wählerprofil erstellt und auf einen Wähler (oder eine Gruppe von Wählern) geschlossen werden kann, weil durch die berechneten  $\beta_i$ -Werte ein Wähler verfolgt werden kann.

---

<sup>1</sup>Der heutige Gegner kann nur jene Algorithmen und Funktionen effizient berechnen, welche maximal ein polynominaler Aufwand  $O(f(x))$  besitzen.

**Minimal Trust** Das Protokoll beinhaltet weniger starke Vertrauensannahmen als die heute implementierten Wahlsysteme. Heute müssen trusted parties verschiedene Tätigkeiten, wie das Anonymisieren der Stimmen durch Vermischen oder das Entschlüsseln von Stimmen, vornehmen und haben somit einen grossen Einfluss auf den korrekten Ablauf einer Wahl.

Im E-Voting Protokoll wird die Vertrauensannahme minimiert, indem bis auf die Private Voter Credentials der Wähler alle Informationen der Wahl öffentlich zugänglich gemacht werden. Somit kann der gesamte Prozess von jedem nachvollzogen und kontrolliert werden. Die Sicherstellung des korrekten Ablaufs liegt also nicht mehr in der Verantwortung einer einzelnen Partei, sondern kann von beliebig vielen unabhängigen Parteien individuell kontrolliert werden.

### 3.3. Tor als möglicher alterner Kanal

Im Rahmen des Ziels „Teil 2a: Recherche Anonymer Kanal“ (gemäss Kapitel 2.1) wurde das Anonymisierungsnetzwerk Tor als möglicher Kanal angeschaut. Bei Tor handelt es sich um ein von Freiwilligen betriebenes und vorangetriebenes Projekt, welches z.B. von Electronic Frontier Foundation (EFF) für die Gewährleistung der Privatsphäre und Anonymität empfohlen wird [7]. Die Website des Tor-Projekts [8] enthält neben Anwendungsbeispielen auch weiterführende technische Informationen zu Tor. Im folgenden Abschnitt werden nur die für diese Arbeit relevanten Punkte von Tor adressiert.

#### 3.3.1. Funktionsweise von Tor

Das Tor-Netzwerk besteht aus verschiedenen Knoten (engl. Nodes) und einem Verzeichnisserver. Die Abbildung 3.1 zeigt diesen Aufbau. Damit der Client (Alice) eine Anfrage an einen Server (Bob) senden kann, fragt die Client-Software eine Liste aller nutzbaren Tor-Knoten beim Verzeichnisserver an. Anschliessend baut die Client-Software über drei zufällig gewählte Tor-Knoten eine Verbindung (Circuit) zum Ziel auf, welche alle 10 Minuten geändert wird.

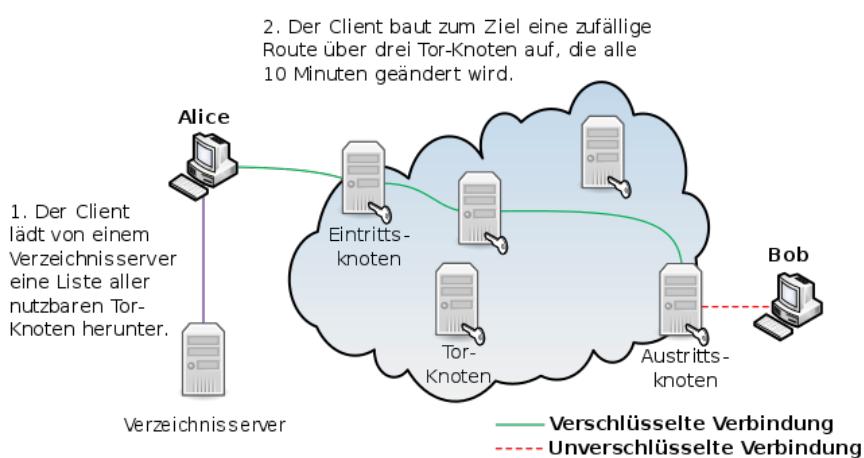


Abbildung 3.1.: Arbeitsweise des Tor-Netzwerks aus [10]

Der Datenverkehr zwischen dem Client und dem Eintrittsknoten und innerhalb des Tor-Netzwerks ist verschlüsselt. Allerdings endet dieser verschlüsselte Verkehr beim Austritt aus dem Tor-Netzwerk, beim Austrittsknoten. Wenn der Server seine Ressourcen unverschlüsselt zur Verfügung stellt (zum Beispiel über http), kann der Austrittsknoten (engl. Exit Node) sämtlichen Verkehr mitlesen und manipulieren. Darum ist es zwingend notwendig, dass die Ressourcen über eine sichere Verbindung angeboten werden, beispielsweise über eine SSL-/TLS-gesicherte Verbindung (über https).

### 3.3.2. Sicherheitsaspekte von Tor

Die Verbindungen über das Tor-Netzwerk sind anonymisiert, weil jede Drittpartei nur beschränktes Wissen hat. Tor basiert auf dem Konzept des „Onion Routing“. Der Begriff Onion (Zwiebel) beschreibt den schichtweisen Aufbau der Kommunikation. Der Client (Alice) verschlüsselt die Nachricht mehrfach, so dass diese nur in der definierten Reihenfolge von den vorgegebenen Drittparteien entschlüsselt werden kann. Es handelt sich um ein asymmetrisches Verschlüsselungsverfahren, bei welchem der Client die entsprechenden Public Keys vom Verzeichnisdienst bezieht.

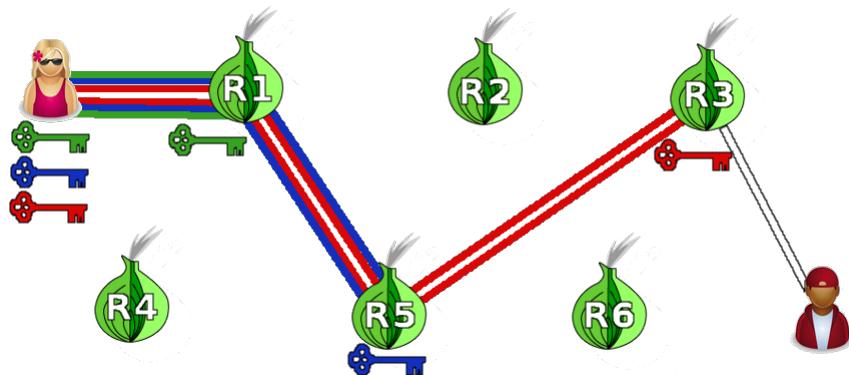


Abbildung 3.2.: Aufbau des Onion Routing aus [4]

Die Abbildung 3.2 zeigt, dass der Eintrittsknoten  $R_1$  nur einen ersten Teil entschlüsseln kann, sozusagen die erste Schale. Er kann erst anschliessend der Nachricht entnehmen, zu welchem Knoten er die Nachricht weiterleiten muss. Der nächste Knoten  $R_5$  kann dann den für ihn bestimmten Teil entschlüsseln und erst der Austrittsknoten  $R_3$  sieht die Nachricht im Klartext. Diese Klartextnachricht wird danach vom Austrittsknoten an das eigentliche Ziel gesendet.

### 3.3.3. Einbindung ins Protokoll

Das Protokoll sieht bereits eine Stimmabgabe über einen anonymen Kanal vor, ohne diesen jedoch genauer zu definieren. In einer ersten Version des Proof of Concept wurde die Stimmabgabe lediglich über einen TLS-verschlüsselten Kanal zum Bulletin-Board gesendet. Ein Angreifer ist nicht in der Lage den Inhalt des Ballots einzusehen, da er den verschlüsselten Kanal nicht einsehen kann. Es besteht jedoch die Möglichkeit, dass der Angreifer feststellen kann, dass der Wähler Verbindung mit dem Bulletin-Board hergestellt hat. Zusammen mit dem Eintreffen resp. der Publikation einer Stimme auf dem Bulletin-Board sind Rückschlüsse über die Herkunft der Stimme möglich.

Zur Sicherstellung der Anonymität und als möglicher anonymer Kanal wurde daher Tor in das Protokoll integriert. Die Stimme wird von der Wahlsoftware an einen lokalen Tor-Proxy-Dienst gesendet, welcher die Kommunikation mit dem Tor-Netzwerk übernimmt. Dieser Proxy-Dienst baut anschliessend den Tor-Circuit auf, analog der obenstehenden Einführung zu Tor. Da das Bulletin-Board ausschliesslich über eine verschlüsselte HTTP-Verbindung ([https](https://)) erreichbar ist, ist die Nachricht *end-to-end* verschlüsselt. Diese verschlüsselte Übertragung lässt den Exit-Nodes keine Möglichkeit Daten einzusehen.



## 4. Produktbeschreibung

Ziel der Arbeit war die Erarbeitung eines Proof of Concepts für ein System zum Erstellen und Durchführen von elektronischen Wahlen basierend auf dem E-Voting Protokoll von Haenni und Locher [2]. Das folgende Kapitel befasst sich mit der Planung und dem Design dieses Proof of Concepts.

### 4.1. Use Cases

Für die Analyse der nötigen Anforderungen an das Produkt, wurden in einem ersten Schritt Use Cases definiert und ausgearbeitet. Dabei wurden folgende vier Use Cases verwendet, welche in ausgearbeiteter Form im Anhang B der Arbeit abgedruckt sind.

**1: Wähler Registrierung** Der erste Use Case zur „Wähler Registrierung“ beschreibt wie ein Wähler neue Private Credentials  $(\alpha, \beta)$  für sich erstellen kann und anschliessend das dazugehörige Public Credential  $u$  auf dem Bulletin-Board registriert. Damit ist der Wähler nach Abschluss dieses Prozesses registriert und kann nun für zukünftige Wahlen berücksichtigt werden.

**2: Wahl Definition** Der zweite Use Case „Wahl Definition“ befasst sich mit der Erstellung einer Wahl. Hier erstellt ein Wahladministrator eine neue Wahl. Er bestimmt über was abgestimmt wird und welche Wähler an dieser Wahl teilnehmen dürfen. Danach wird lokal ein neues Election Credential  $\hat{h}$  bestimmt. Zusätzlich werden aus den Public Credentials  $u$  der ausgewählten Wählern  $U$  die Koeffizienten  $A$  für das Polynom  $P$  berechnet. Abschliessend werden die Werte  $(\hat{h}, U, A)$  der Wahl auf dem Bulletin-Board publiziert.

**3: Stimmabgabe** Beim dritten Use Case „Stimmabgabe“ selektiert der Wähler eine laufende Wahl und gibt seine Stimme dafür ab. Anschliessend werden mit der Stimme  $e$  und den Private Credentials  $(\alpha, \beta)$  des Wählers die kryptographischen Größen  $c, d, \pi_1, \pi_2, \pi_3, \hat{u}$  berechnet und als Stimmzettel  $(c, d, e, \pi_1, \pi_2, \pi_3, \hat{u})$  auf dem Bulletin-Board publiziert.

**4: Auszählung – Verifikation** Beim Use Case „Auszählung – Verifikation“ wird von einem Verifier eine abgeschlossene Wahl ausgezählt. Es werden alle Stimmzettel, welche zu dieser Wahl abgegeben wurden, auf ihre Gültigkeit geprüft. Dabei wird in einem ersten Schritt geprüft, ob die Stimmzettel innerhalb des definierten Zeitintervalls der Wahl eingetroffen sind. In einem zweiten Schritt werden die kryptographischen Beweise  $\pi_1, \pi_2, \pi_3$  auf ihre Korrektheit geprüft. Sind nach diesem Schritt pro Wähler noch mehrere gültige Stimmzettel vorhanden, so wird nach einer definierten Strategie einer der Stimmzettel ausgewählt. Für diesen Prototyp wurde definiert, dass immer der älteste valide Stimmzettel selektiert wird. Aus den so erhaltenen gültigen und selektierten Stimmen wird anschliessend das Resultat der Wahl berechnet. Sollte einer der selektierten Stimmzettel keine gültige Stimme enthalten, zum Beispiel eine leere Stimme, so wird diese für das Resultat nicht berücksichtigt. Es wird aber auch kein neuer Stimmzettel desselben Wählers gewählt. Abschliessend kann das erhaltene Resultat vom Verifier auf dem Bulletin-Board publiziert werden.

## 4.2. High-Level Aufbau

Aus den Use Cases sind drei verschiedene Akteure erkennbar, welche mit dem Wahlsystem arbeiten. Ein Wahladministrator, welcher die Wahlen definiert, ein Wähler, welcher sich registriert und seine Stimme abgibt, und schliesslich der Verifier, welcher eine Wahl verifiziert und auszählt. Alle drei Rollen interagieren dabei indirekt miteinander, indem sie Daten vom Bulletin-Board beziehen und darauf ablegen.

Um dieses Verhalten widerzuspiegeln, wurde das Wahlsystem in vier separate Teilprogramme aufgeteilt. Die AdminApp, die VoterApp, die VerifierApp und das Bulletin-Board. Diese Aufteilung hat zusätzlich zum Ziel, dass die einzelnen Teilprogramme mit geringerem Aufwand erweitert oder gar ersetzt werden können. Um die Verantwortungen und Anforderungen der einzelnen Teile des Wahlsystems zu definieren, können wiederum die Resultate aus den Use Cases herbeigezogen werden.

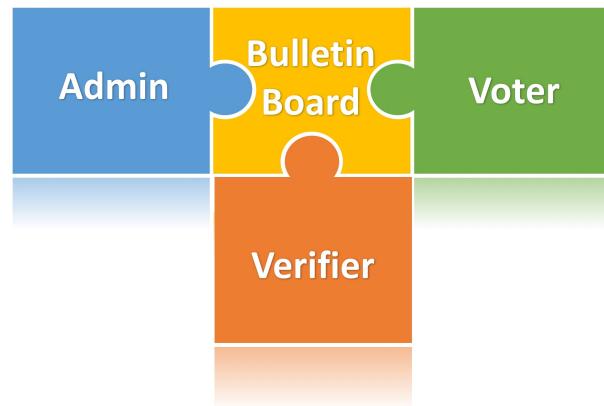


Abbildung 4.1.: Komponenten abcVote schematisch dargestellt

Die detaillierten Anforderungen an die verschiedenen Komponenten sind im Anhang C aufgelistet. Im Überblick kann festgehalten werden, dass die AdminApp in Zusammenarbeit mit dem Bulletin-Board den Use Case „Wahl Definition“ abbildet und einem Benutzer ermöglicht, eine neue Wahl zu definieren und zu publizieren. Die VoterApp deckt die beiden Use Cases „Wähler Registrierung“ und „Stimmabgabe“ ab. Die VerifierApp wird für das Auszählen einer Wahl verwendet und ist somit für den Use Case „Auszählung-Verifikation“ zuständig. Das Bulletin-Board dient als zentrale Datenablage für das Wahlsystem und ist somit das Verbindungsstück zwischen den drei übrigen Komponenten. Die nachfolgende Abbildung 4.2 zeigt einen ersten Entwurf der Schnittstellen zwischen Apps und Bulletin-Board.

## 4.3. Technologiewahl

Die erarbeiteten Anforderungen an die einzelnen Komponenten bestimmen die verwendeten Technologien. Neben den Anforderungen aus den Use Cases waren für die Technologiewahl zusätzlich die Rahmenbedingungen der Arbeit, sowie die Erkenntnisse aus dem Vorprojekt ausschlaggebend.

Eine der wichtigsten Punkte für die Technologiewahl bildet hier das E-Voting-Protokoll. Um die kryptographischen Beweise des Protokolls zu berechnen, wurde die BFH-eigene Library *Unicrypt* verwendet. Da die Unicrypt-Library in Java geschrieben wurde, folgt daraus, dass auch alle Programmteile, welche Unicrypt verwenden auf Java zurückgreifen sollten. Im Wahlsystem abcVote wurde daher für die Teilprogramme AdminApp, VoterApp und VerifierApp Java als Programmiersprache verwendet. Als Framework für die graphische Oberfläche wurde JavaFX verwendet. Zur Verwaltung der Abhängigkeiten wurden AdminApp, VoterApp und VerifierApp als Maven-Projekte aufgebaut.

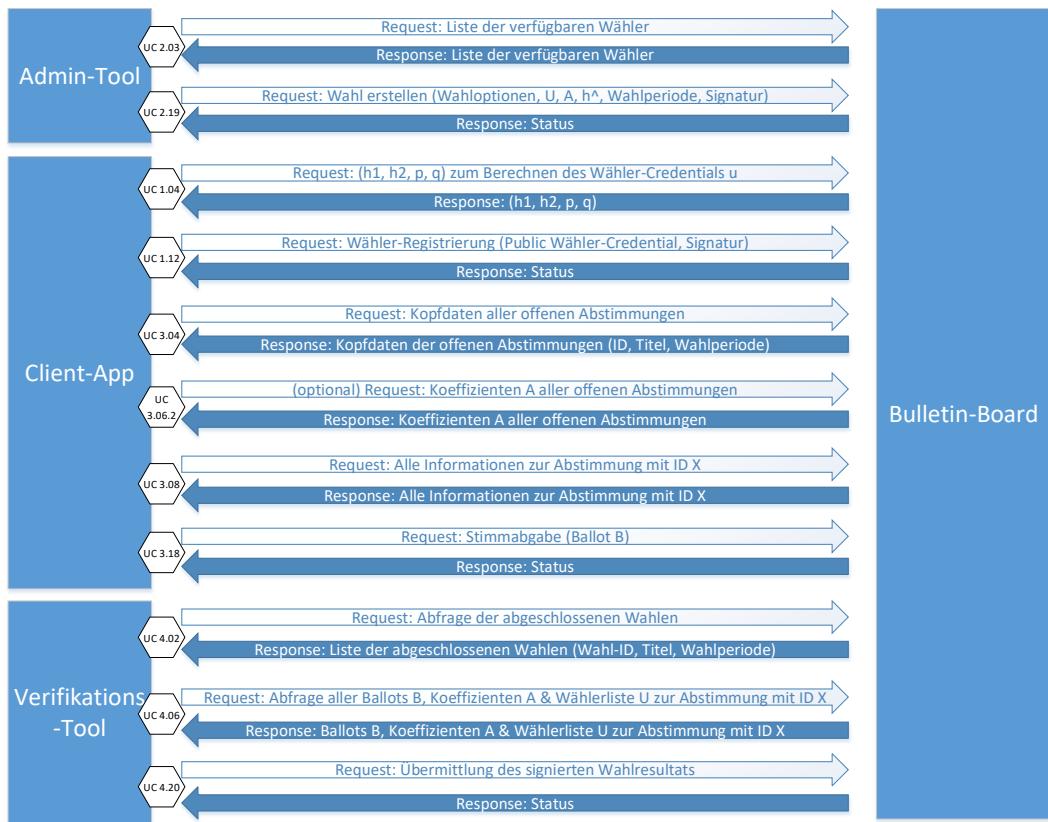


Abbildung 4.2.: Kommunikation innerhalb abcVote

Für das Bulletin-Board hingegen gelten andere Anforderungen. Wie bereits erwähnt ist die Hauptaufgabe des Bulletin-Boards das Ablegen und Wiedergeben der Daten für das Wahlsystem. Insbesondere werden hier keine kryptographischen Beweise bezüglich dem E-Voting Protokoll berechnet. Daher ist das Bulletin-Board unabhängig von der Unicrypt-Library. Des Weiteren wurde bei der Technologiewahl für das Bulletin-Board darauf geachtet, dass es möglichst einfach zu gestalten ist. Grund dafür ist, dass aufgrund eines Kann-Zieles dieser Arbeit das Bulletin-Board eventuell durch eine mit „Append-only“-Eigenschaft ersetzt wird.

Basierend auf diesen Grundlagen wurde entschieden hier auf bereits bekannte Technologien zurückzugreifen und das Bulletin-Board als REST-Schnittstelle mit PHP zu entwickeln. Zur Datenablage wurde dabei eine MySQL-Datenbank verwendet. Um sicherzustellen, dass während der Entwicklung alle Teilprogramme jeweils mit der aktuellsten Version des Bulletin-Boards kompatibel sind, wurde für die Entwicklung des Wahlsystems ein Apache-Webserver eingerichtet, auf welchem stets die aktuellste funktionierende Version des Bulletin-Boards veröffentlicht wurde. Im Anhang D wird beschrieben, wie man den Webserver und die Java-Applikationen einrichtet.

Nachfolgend sind die wichtigsten Technologien für das Wahlsystem nochmals tabellarisch zusammengefasst.

Technologie	Zweck
Netbeans	Entwicklungsumgebung (IDE)
Java 8 mit JavaFX	Applikationen und graphische Benutzeroberflächen (GUI)
Github	Source Code Verwaltung
Unicrypt	Library mit kryptographischen Funktionen
JSON	Datenaustausch zwischen den Komponenten
JWS	Teil des JOSE-Frameworks zur Signierung von JSONs
LAMP-Stack	Apache mit PHP und MySQL für Bulletin-Board
Slim	PHP-Framework als Basis für REST-Funktionalität des Bulletin-Boards

Tabelle 4.1.: Technologiewahl

## 5. Detaillierter Aufbau

Basierend auf den bisher erarbeiteten Anforderungen wurde die Architektur des Endproduktes abgebildet. Das Wahlsystem für den Proof of Concept basiert auf den bisher erarbeiteten Komponenten: Bulletin-Board, AdminApp, VoterApp und VerifierApp. Zusätzlich wurde eine fünfte Komponente definiert, das *util*-Package. Dieses fungiert als Hilfs-Package, damit die Klassen-Definitionen der verschiedenen Elemente aus dem E-Voting Protokoll zentral definiert werden können. In den folgenden Abschnitten wird genauer auf den Aufbau und die Kompetenzen der einzelnen Teile eingegangen.

Zuerst aber müssen hier noch einige Aspekte erwähnt werden, welche einen allgemeinen Einfluss auf die Definition der Komponenten hatten. Der wichtigste Einfluss auf das Design des Projekts ist der Umstand, dass als Endprodukt ein Proof of Concept angestrebt wurde.

Dies bedeutet, dass der Fokus stark auf die korrekte Umsetzung des E-Voting Protokolls gesetzt wurde und weniger auf die komplette Abdeckung aller Wahltypen und Komfortfunktionen für den Benutzer. Dies hat zur Folge, dass für das Endprodukt nur Wahlen durchgeführt werden können, bei denen aus  $n$  Möglichkeiten  $k$  Elemente ausgewählt werden können. Andere Abstimmungen wie zum Beispiel Parlamentswahlen wurden nicht umgesetzt. Trotzdem wurde beim Design darauf geachtet, dass genügend Spielraum vorhanden ist, um diese zu einem späteren Zeitpunkt zu integrieren.

Des Weiteren wurde im Rahmen des Proof of Concepts definiert, dass die Benutzer authentisiert werden, indem sie den Inhalt ihrer POST-Requests an das Bulletin-Board signieren. Die einzige Ausnahme bildet hier die Abgabe des Stimmzettels, welche anonym durchgeführt werden muss. Um die Korrektheit der Signaturen zu prüfen wird auf dem Bulletin-Board eine PKI hinterlegt, welche die Zertifikate aller zugelassenen Benutzer verwaltet. In Absprache mit den Betreuern wurde definiert, dass eine solche PKI als gegeben vorausgesetzt werden kann. Deshalb besitzt das Bulletin-Board eine Liste mit zugelassenen Zertifikaten. Allerdings beinhaltet das Wahlsystem keine Instrumente für die Verwaltung dieser Liste.

### 5.1. Bulletin-Board

Die Hauptrolle des Bulletin-Board ist und bleibt die zentrale Datenablage des Wahlsystems. Obwohl bei der Technologiewahl entschieden wurde, eine MySQL-Datenbank für die Verwaltung der Daten zu benutzen, wird hier bewusst auf die Nutzung der Update- und Delete-Funktionen von MySQL verzichtet. Diese Einschränkung wird definiert, um so ein „append-only“-Bulletin-Board zu simulieren, wie dies im E-Voting Protokoll von Haenni und Locher gefordert wird. Daraus folgt, dass auch die REST-Schnittstellen des Bulletin-Boards auf POST-Requests zum Hinzufügen neuer Datensätze und GET-Requests zum Abfragen von Daten beschränkt werden.

**Aufbau** Das Bulletin-Board basiert auf dem *Slim* PHP-Framework und verwendet *Composer* für die Verwaltung der Abhängigkeiten. Das Framework bietet eine Basis zur Erstellung von Web-Apps mit REST-Funktionalität. Das Herzstück von Slim (und somit auch des Bulletin-Boards) ist die Datei *index.php*. Sie enthält für jeden Aufruf eine sogenannte *Route*, an welche die entsprechenden Daten weitergegeben werden. Die Datei ist ausführlich dokumentiert und enthält Informationen darüber, für welchen Use Case, welche Route verwendet wird. Neben dieser Kernkomponente wurde ein objekt-orientierter Ansatz zur

Realisierung der protokoll-spezifischen Aspekte verwendet. Im Ordner *classes* sind für jedes Objekt eine *Entity*-Klasse und eine *Mapper*-Klasse vorhanden. Erstere definiert den Aufbau eines Objekts und bietet mögliche Funktionsaufrufe. Letztere dient zur Übersetzung zwischen den Objekten und der Datenbank.

Ausserdem bietet das Bulletin-Board die Möglichkeit die in der Datenbank abgespeicherten Daten als Webseite anzuseigen. Für jede solche Webseite gibt es eine Template-Datei im Ordner *templates*. Folgende Ansichten wurden realisiert: *ballots* (Anzeige der gespeicherten Ballots), *elections* (Auflistung der vorhandenen Wahlen), *parameters* (Parameter dieses Bulletin-Boards), *results* (Liste der publizierten Wahlresultate) und *voters* (Auflistung der registrierten Wähler). Diese Webseiten sind über die Bulletin-Board Web-Adresse gefolgt von `/view/(template-name)` erreichbar.

**Kommunikation** Unter Berücksichtigung der Resultate aus der Analyse der Use Cases wurden die benötigten Kommunikationsabläufe als REST-Schnittstellen definiert. Die Definitionen der Schnittstellen und die Struktur der verschickten JSON-Daten wurden vor der Implementierungsphase definiert, damit das Bulletin-Board und die übrigen Komponenten parallel zueinander entwickelt werden konnten.



Abbildung 5.1.: REST-Schnittstellen des Bulletin-Boards mit entsprechenden HTTP-Befehlen (GET, POST)

Während die Struktur der JSON-Daten für die übrigen Applikationen im Wahlsystem genau definiert werden muss, müssen auf der Seite des Bulletin-Boards nur gewisse Grundstrukturen der JSON-Daten eingehalten werden. Ein Auslesen der JSON-Strings ist auf dem Bulletin-Board nur notwendig, wenn eine Eigenschaft des Objekts separat in der Datenbank abgelegt werden muss. Zum Beispiel, wenn nach diesem Merkmal gefiltert werden soll. Dies bedeutet, dass bezüglich dem Bulletin-Board nur diese Eigenschaften einen festen Platz in der JSON-Struktur benötigen und somit die übrige Struktur beliebig angepasst und erweitert werden kann. Im Anhang E sind JSON-Beispiele für die verschiedenen Übertragungen abgedruckt.

Die JSON-Strukturen werden in den jeweiligen Java-Anwendungen signiert und anschliessend als JWS-Daten an das Bulletin-Board gesendet. Einzig die Stimmabgabe erfolgt unsigniert, da sonst die Anonymität nicht mehr gegeben ist. JWS (JSON Web Signature) ist Teil des JOSE<sup>1</sup>-Frameworks und ist im RFC 7515 spezifiziert. Eine JWS-Signatur besitzt einen Header, einen Payload und eine Signatur. Diese Teile sind jeweils mit „base64url“ encodiert und durch Punkte getrennt aneinander konkateniert.

**Datenbank** Damit dieser flexible Aufbau in Zukunft genutzt werden kann, um das System zu erweitern und anzupassen, wurde beim Design der Datenbank darauf verzichtet für jede Eigenschaft eines Objekts eine separate Spalte zu definieren. Stattdessen wird der JSON-String als Ganzes in der Datenbank abgelegt. Nur wenn eine Eigenschaft für eine Suche oder Filterfunktion verwendet wird, wird diese aus dem JSON-Objekt ausgelesen und separat abgelegt.

Die Datenbank besteht aus sechs Tabellen und zwei Abhängigkeiten, was das nachfolgende ERM-Diagramm grafisch darstellt. Die Wähler werden in der Tabelle *tbl\_voters*, Parameter in der Tabelle *tbl\_parameters* und die Zertifikate der PKI-Simulation in *tbl\_certificates* abgelegt. Jede Wahl wird in einem Eintrag in der Tabelle *tbl\_elections* abgelegt. Der verwendete Primary Key „id“ der Tabelle wird als Foreign Key der beiden folgenden Tabellen verwendet, welche die abgegebenen Stimmen enthält, und *tbl\_results*, welche die verifizierten Resultate der Wahlen beinhaltet.

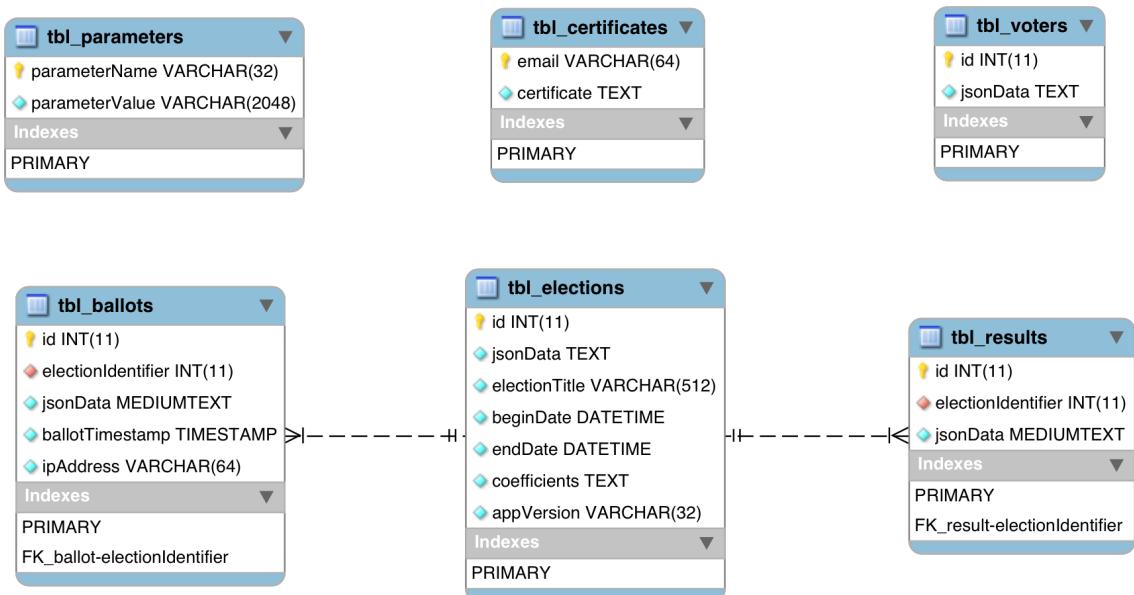


Abbildung 5.2.: ERM-Diagramm der abcVote Datenbank

## 5.2. Util-Package

Das Util-Package des Wahlsystems „abcVote“ enthält die Klassen-Definitionen aller Objekte, welche für die Durchführung einer Wahl benötigt werden. Zusätzlich sind in diesem Projekt Controller-Klassen enthalten, welche von allen Komponenten gemeinsam verwendet werden. Im Folgenden wird auf die einzelnen Klassen- und Controller-Definitionen im Detail eingegangen.

<sup>1</sup> JOSE ist die Abkürzung für „JSON Object Signing and Encryption“

### 5.2.1. Model-Klassen

Model-Klassen sind Klassen, um die verschiedenen Objekte einer Wahl zu definieren. Die Klassen beinhalten sowohl die kryptographischen Größen des E-Voting Protokolls sowie die Definitionen für die Wahl selbst. Im Folgenden werden die Klassen beschrieben und ihre Rolle im Protokoll und bezüglich der Wahl geklärt.

**Parameters** Diese Klasse beinhaltet alle kryptographischen Größen, welche global für das ganze Wahl- system gelten und zur Berechnung aller kryptographischen Größen des Protokolls verwendet werden. Insbesondere sind hier die Zyklischen Gruppen für das Protokoll  $G_p$  und  $G_q$  definiert. Zudem werden die Generatoren  $h_0, h_1, h_2$  der Gruppe  $G_p$  und die Generatoren  $g_0, g_1$  der Gruppe  $G_q$  festgelegt. Die beiden Generatoren  $g_0, g_1$  werden im Protokoll verwendet, um aus den Private Credentials eines Wählers das Public Credential zu berechnen. Gleichzeitig dienen die Generatoren  $h_0, h_1, h_2$  zur Berechnung des Commitments  $d$ . Analog dazu werden die Generatoren  $g_0, g_1$  für die Berechnung des Commitments  $c$  verwendet. Zusätzlich können zur Vereinfachung in dieser Klasse zusätzlich die Gruppen  $\mathbb{Z}_p, \mathbb{Z}_q$  und Schermen zur Berechnung der beiden Commitments  $c$  und  $d$  abgerufen werden. Diese Elemente könnten zwar von den obengenannten Größen abgeleitet werden, wurden hier aber auch integriert, um die Berechnung der Beweise zu vereinfachen.

**Voter** Die Voter-Klasse repräsentiert einen registrierten Wähler und beinhaltet drei Eigenschaften: Email-Adresse, Public Credential und App-Version. Dabei entspricht das Public Credential der Größe  $u$  aus dem E-Voting Protokoll. Die E-Mail-Adresse repräsentiert den Wähler selbst. In einer späteren Version könnten zusätzlich zur E-Mail-Adresse natürlich noch weitere Personalien des Wählers hier hinterlegt werden, welche für Such- und Filterfunktionen genutzt werden könnten.

**Private Credentials** Diese Klasse beinhaltet die Private Credentials eines Wählers  $\alpha, \beta$ . Zusätzlich kann aufgrund der Private Credentials das Public Credential  $u$  berechnet und ausgegeben werden. Die Klasse benötigt dafür die Parameter des Wahlsystems in Form eines Parameter-Objekts. Sie bietet zum einen die Möglichkeit an, basierend auf den Parametern des Wahlsystems neue Private Credentials zu erstellen oder zum anderen bestehende Werte wiederherzustellen.

**Election** Die Election-Klasse beinhaltet die Definitionen der eigentlichen Wahl. Für die Wahl werden hier die Rahmenbedingungen definiert. Die Election-Klasse enthält eine Liste von Voter-Objekten. Diese repräsentiert die Liste  $U$  aller zugelassenen Wähler. Außerdem wird in dieser Klasse ein Start- und Enddatum definiert. Während diesem Intervall gilt die Wahl als offen und es können Stimmen abgegeben werden. Der eigentliche Inhalt der Wahl, also über was genau abgestimmt wird, wird in einem ElectionTopic-Objekt auf der Election-Klasse hinterlegt. Auf den Aufbau dieses Objekts wird im nächsten Abschnitt genauer eingegangen. Neben den Definitionen der Wahl enthält auch diese Klasse protokoll- spezifische Größen. Zum einen ist dies das Election Credential  $\hat{h}$ , zum anderen das Polynom  $P$ , welches aus den Public Credentials der zugelassenen Wähler berechnet wird.

**ElectionTopic** In der ElectionTopic-Klasse wird gespeichert über was genau abgestimmt wird. Darin enthalten ist eine Frage oder eine Bezeichnung, welche festlegt über was genau abgestimmt wird. In einer dazugehörigen Liste werden die Auswahlmöglichkeiten zum Thema festgelegt. Zusätzlich kann festgelegt werden, wie viele der Auswahlmöglichkeiten bei der Abstimmung ausgewählt werden müssen. Mit der aktuellen Umsetzung der Klasse lassen sich zurzeit nur Wahlen abbilden, bei denen aus  $n$  Möglichkeiten  $k$  Elemente ausgewählt werden. Weitere Wahltypen, wie zum Beispiel Parlamentswahlen, lassen sich

noch nicht abbilden. Diese Klasse ist jedoch unabhängig vom E-Voting Protokoll. Somit können weitere Wahltypen integriert werden, ohne dass der E-Voting-Protokoll-Teil des Wahlsystems angepasst werden muss.

**ElectionHeader** Die ElectionHeader-Klasse wird als Hilfsklasse bei der Auflistung von Wahlen verwendet. Sie beinhaltet nur die wichtigsten Kopfdaten einer Wahl, namentlich Id, Titel und Abstimmungsintervall. Somit müssen bei einer Auflistung der Wahlen nicht die kompletten Daten jeder Wahl vom Bulletin-Board geholt werden.

**Ballot** Die Ballot-Klasse repräsentiert einen elektronischen Stimmzettel. Dieser Stimmzettel ist an eine Wahl gebunden, indem ein Election-Objekt hinterlegt wird. Wie ein Papierstimmzettel enthält auch diese Klasse die Auswahlmöglichkeiten, für welche sich ein Wähler entschieden hat. Gleichzeitig implementiert die Ballot-Klasse auch das Ballot wie es im E-Voting Protokoll definiert wurde. Somit sind hier auch die Commitments  $c$ ,  $d$  und der Beweise  $\pi_1$ ,  $\pi_2$ ,  $\pi_3$  hinterlegt, welche zum Protokoll gehören. Um diese Beweise zu berechnen, müssen in einem ersten Schritt die Wahl und die selektierten Optionen festgelegt werden. Danach werden dem Ballot-Objekt die Private Credentials des abstimmenden Wählers übergeben. Anschliessend werden die Beweise in Abhängigkeit von den selektierten Optionen  $e$  und des Election Credentials  $\hat{h}$  berechnet. Für die Auszählung der Wahlzettel wird zudem ein Timestamp definiert. Er hält den Zeitpunkt fest, an welchem der Wahlzettel auf dem Bulletin-Board eingetroffen ist. Zusätzlich wird ein Flag definiert, um Ballots zu markieren, welche während dem Auszähl-Prozess eine Validierung nicht bestanden haben.

**ElectionResult** Die ElectionResult-Klasse beinhaltet das Abstimmungsresultat einer Wahl. Ein Abstimmungsresultat ist immer an eine Wahl (Election-Objekt) gebunden. Deshalb wird auf dem ElectionResult-Objekt eine Wahl hinterlegt. Zusätzlich zum eigentlichen Resultat werden auch die validierten Ballots hier abgelegt. Somit kann das Resultat anhand der ausgewerteten Stimmzettel nachvollzogen werden.

### 5.2.2. Controller-Klassen

Die Controller-Klassen decken Aktivitäten ab, welche von allen drei Programmen AdminApp, VoterApp und VerifierApp genutzt werden. Somit kann die gleiche Programmlogik mehrfach eingesetzt werden.

**CommunicationController** Der CommunicationController regelt die Kommunikation zwischen Apps und Bulletin-Board. Bei seiner Initialisierung kann die Adresse des zu verwendenden Bulletin-Boards übergeben werden. Der CommunicationController dient als Verbindungsstück zwischen den Apps und dem Bulletin-Board. Dabei werden alle Objekte, welche an das Bulletin-Board übertragen werden müssen, vom CommunicationController in JSON-Objekte übersetzt und signiert, bevor sie an die REST-Schnittstelle geschickt werden. Die einzige Ausnahme bildet hier die Übertragung von Ballots, welche bei der Übertragung nicht signiert werden. Ausserdem werden auch die GET-Anfragen an das Bulletin-Board über den CommunicationController abgesetzt. Der erhaltene JSON-String wird anschliessend vom Controller in die entsprechenden Objekte der Model-Klassen übersetzt und dem Aufrufer zurückgegeben.

**KeyStoreController** Der KeyStoreController verwaltet den Zugriff auf den lokalen Java-KeyStore. Im KeyStore ist ein Zertifikat für das Signieren der versendeten JSON-Strings abgelegt. Zusätzlich dient der KeyStore auch als Lagerort für die Private Credentials, welche in der VoterApp erstellt werden. Über den KeyStoreController können Zertifikat und Private Credentials abgerufen oder neu generierte Private Credentials im KeyStore abgelegt werden.

**SignatureController** Der SignatureController ist für die Signierung der JSON-Strings zuständig, welche an das Bulletin-Board gesendet werden. Der SignatureController nimmt ein JSON-String entgegen und signiert diesen mit dem Zertifikat, welches im Keystore abgelegt wurde. Das daraus resultierende JWS (JSON Web Signature) kann anschliessend an das Bulletin-Board gesendet werden.

### 5.3. Architektur der Java-Applikationen

Bevor auf die spezifischen Eigenschaften und Verantwortungen der einzelnen Applikationen eingegangen wird, werden hier die architektonischen Aspekte behandelt, welche die drei Applikationen gemeinsam haben. Da die drei Applikationen AdminApp, VoterApp und VerifierApp Teil des gleichen Wahlsystems sind und mit den gleichen Technologien arbeiten, wurde entschieden, für alle drei Projekte dasselbe Schema zu verwenden.

Als Vorlage für den Aufbau der JavaFX-Apps wurde eine Variante des MVC Design-Patterns verwendet (siehe Abbildung 5.3). Für jedes Anzeigefenster wird eine FXML-Datei für die Definition der Benutzeroberfläche und ein dazugehöriger Controller erstellt. Die Controller-Klasse verwaltet die Benutzereingaben mit Hilfe der Model-Klassen. Die Model-Klassen werden ausserhalb der einzelnen Projekte in der util-Library definiert.

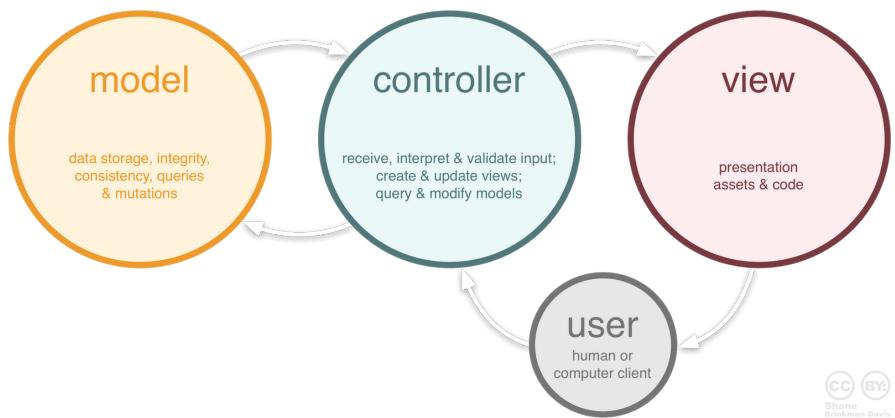


Abbildung 5.3.: MVC Design-Pattern aus [9]

Damit der Programmablauf zwischen den einzelnen Anzeigefenstern geregelt werden kann, wird eine MainController-Klasse verwendet. Diese verwaltet die Kommunikation zwischen den einzelnen Controllern. Somit können die Controller mithilfe des MainController Objekte an den nächsten Controller im Ablauf übergeben. Zudem sorgt der MainController beim Übergang von einem Anzeigefenster zum nächsten dafür, dass die Benutzeroberfläche entsprechend angepasst wird. Die zweite Rolle des MainControllers ist die Kommunikation zwischen den Controllern der Benutzeroberfläche und den Klassen, welche mit den verschiedenen Datenablagen kommunizieren. Dies hat den Vorteil, dass zum einen die Programm-Logik unabhängig ist von der Ablage der Daten und zum anderen kann so die Kommunikation mit den Datenablagen zentral und einheitlich in der util-Library definiert werden.

Die App-Version wurde in die Applikationen und Kommunikation integriert, um in einem späteren Release einen Anhaltspunkt für Kompatibilitätsprüfungen zu geben. Allerdings wurde diese Funktionalität nicht fertig realisiert und nur in den JSON- und Datenbank-Strukturen implementiert. Mithilfe der App-Version kann erkannt werden, mit welcher Version einer App ein Wähler registriert wurde oder eine Wahl erstellt wurde. So kann mit der App-Version in einem späteren Ausbau verhindert werden, dass veraltete Apps auf Ressourcen zugreifen können, welche mit einer neueren App-Version erstellt wurden. Dies erlaubt Änderungen an der JSON-Struktur, ohne dass das Wahlsystem instabil wird.

## 5.4. AdminApp

Die AdminApp wird vom Wahladministrator verwendet, um eine neue Wahl zu erstellen. Dabei wird der Benutzer Schritt für Schritt aufgefordert, die Definitionen der Wahl einzugeben. Die eingegebenen Wahldefinitionen werden in einem Election-Objekt eingetragen. Dieses wird während dem Programmablauf von Controller zu Controller weitergegeben. Es werden dabei der Titel der Abstimmung, die wahlberechtigten Voter, die Abstimmungsfrage und deren Optionen und der Zeitraum für die Stimmenabgabe definiert. Erst vor der Übertragung der Wahl werden die kryptographischen Größen definiert und berechnet. Zum einen wird der Wahlgenerator  $\hat{h}$  aufgrund der Parameter des Wahlsystems zufällig gewählt. Die Parameter des Wahlsystems werden im Vorfeld auf dem Bulletin-Board abgefragt. Zum anderen wird aus den Public Credentials der ausgewählten Wähler das Polynom  $P$  hergeleitet. Das Polynom  $P$  wird mit der Klasse PolynomialElement von Unicrypt dargestellt. Abschliessend wird das Election-Objekt in ein JSON-Objekt übersetzt. Dabei werden die Unicrypt-Elemente in ihre String-Präsentation übersetzt, damit diese in das JSON-Objekt eingefügt werden können. Das daraus entstehende JSON-Objekt wird signiert und an das Bulletin-Board übertragen.

Damit die Wahldefinition signiert werden kann, wird für den Betrieb der AdminApp ein Signierungszertifikat mit passendem Private Key benötigt. Dieses muss lokal in einem Java-KeyStore abgelegt sein. Zusätzlich muss das Zertifikat in der PKI des Bulletin-Boards hinterlegt werden, damit die Signatur und somit auch die Wahldefinitionen akzeptiert werden. Um auf der Seite des Bulletin-Boards festzustellen, welches Zertifikat aus der PKI verwendet werden soll, wird im JSON-Objekt die E-Mail-Adresse des Zertifikats mitgegeben.

## 5.5. VoterApp

Jeder Wähler verwendet die VoterApp, um sich auf dem Bulletin-Board zu registrieren und seine Stimme für eine Wahl abzugeben. Bei der Registrierung wird der Benutzer aufgefordert, seine E-Mail-Adresse einzutragen. Anschliessend werden mit den Parametern des Wahlsystems zufällig neue Private Credentials  $\alpha, \beta$  gewählt. Aus den Private Credentials wird das Public Credential  $u$  berechnet. Der erhaltene Wert wird nun gemeinsam mit der eingegebenen E-Mail-Adresse in ein JSON-Objekt übersetzt. Das generierte JSON-Objekt wird signiert und an das Bulletin-Board geschickt. Erst nach der Registrierungs-Bestätigung durch das Bulletin-Board legt die VoterApp die neu registrierten Private Credentials lokal im Java-KeyStore ab. So wird sichergestellt, dass die VoterApp nur mit Private Credentials arbeitet, welche auch im Wahlsystem registriert sind.

Für das Signieren am Ende des Registrierungsprozesses wird auch bei der VoterApp ein Signierungszertifikat mit passendem Private Key benötigt. Dieses muss lokal in einem Java-KeyStore abgelegt werden. Zusätzlich muss die auf dem Zertifikat eingetragene E-Mail-Adresse mit der vom Wähler eingegebenen Adresse übereinstimmen. Auch hier gilt, dass das Zertifikat auch in der PKI des Bulletin-Boards hinterlegt sein muss.

**Stimmabgabe** Sobald ein Wähler sich registriert hat, kann er zu einer Wahl seine Stimme abgeben. Dafür werden am Anfang dieses Ablaufs alle zum aktuellen Zeitpunkt laufenden Wahlen vom Bulletin-Board abgefragt und angezeigt. Dabei werden nur die Kopfdaten (Id, Titel, Wahlperiode) der jeweiligen Daten vom Bulletin-Board geschickt, um die übertragene Datenmenge zu reduzieren. Sobald der Wähler eine Wahl ausgewählt hat, zu der er seine Stimme abgeben möchte, werden die gesamten Daten zur Wahl auf dem Bulletin-Board abgefragt. Diese Abfrage wird in zwei Schritten durchgeführt. In einem ersten Schritt werden die Parameter des Wahlsystems abgefragt und lokal gespeichert. In einem zweiten Schritt werden nun die Daten der Wahl abgeholt. Damit diese Daten lokal wieder in ein Election-Objekt

konvertiert werden können, werden nun die Parameter benötigt. Die Daten der Wahl werden anschliessend dargestellt. Nachdem der Wähler die gewünschten Optionen ausgewählt hat, wird daraus der elektronische Stimmzettel als Ballot-Objekt erstellt.

**Ballot-Erstellung** In einem ersten Schritt wird das Ballot erstellt, wobei alle relevanten Daten auf dem Ballot hinterlegt werden. Zum Einen ist dies das Election-Objekt der ausgewählten Wahl, zum anderen eine Liste der selektierten Optionen. Die Parameter des Wahlsystems werden hier von dem Election-Objekt übernommen. Erst zu diesem Zeitpunkt werden die Private Credentials aus dem lokalen KeyStore geladen, damit die kryptographischen Beweise berechnet werden können. Für die Berechnung der Beweise und der dazugehörigen Größen werden Klassen der Unicrypt-Library verwendet. Zuerst werden alle Elemente vorbereitet, welche anschliessend zum Berechnen der Beweise verwendet werden. Dazu gehören Private Credentials  $\alpha$  und  $\beta$ , das Public Credential  $u$ , die Commitments  $c$  und  $d$  sowie die Stimme  $e$ .

**Berechnung der Zero-Knowledge Proofs und Übertragung** Aus den bereits vorhandenen Private Credentials wird  $u$  erneut berechnet. Für die Berechnung der Commitments werden Objekte der Klasse *PedersenCommitmentScheme* verwendet. Diese Commitment-Funktionen werden vom Parameters-Objekt zur Verfügung gestellt. Die Stimme  $e$  wird erstellt, indem alle selektierten Optionen durch Kommas getrennt in einen String geschrieben werden. Dieser String muss zusätzlich in ein StringMonoid-Objekt überführt werden, damit er in die Unicrypt Beweise eingebunden werden kann. Ziel ist es, die Stimme  $e$  in das Challenge der drei Non-interactive Zero Knowledge Proofs einzubinden und diese so von der Stimme abhängig zu machen. Dazu werden FiatShamirSigmaChallengeGenerator-Objekte in Abhängigkeit von  $e$  erstellt. In einem nächsten Schritt werden die Objekte für die Berechnung der drei Beweise definiert. Hier wird für das Generieren von  $\pi_1$  ein *PolynomialMembershipProofSystem*-Objekt verwendet, für  $\pi_2$  ein *DoubleDiscreteLogProofSystem*-Objekt und für  $\pi_3$  ein *EqualityPreimageProofSystem*-Objekt. Beim Erstellen der drei ProofSystems werden die vorbereiteten ChallengeGeneratoren übergeben, um die Abhängigkeit von  $e$  herzustellen. Anschliessend werden mit den ProofSystems die drei Beweise berechnet. Nachdem alle Beweise berechnet wurden, wird das Ballot in ein JSON-Objekt übersetzt und unsigniert an das Bulletin-Board übertragen. Der Stimmzettel wird hier bewusst nicht signiert, da dadurch dieser eindeutig einem Wähler zugewiesen werden könnte.

## 5.6. VerifierApp

Die VerifierApp wird von einem Verifier benutzt, um alle Stimmzettel einer Wahl zu verifizieren und auszuzählen. Das Auszählen einer Wahl stellt keine Anforderungen an den Verifier und kann daher von jedem durchgeführt werden, der in Besitz der VerifierApp ist. Einzig wenn das erhaltene Resultat zum Schluss auf dem Bulletin-Board publiziert werden soll, wird ein Signierungszertifikat mit passendem Private Key benötigt. Dieses Zertifikat muss zusätzlich in der PKI des Bulletin-Boards hinterlegt sein.

Der Beginn des Auszählungsprozesses ähnelt stark dem Prozess der Stimmabgabe in der VoterApp. Die Kopfdaten der bereits abgeschlossenen Wahlen werden vom Bulletin-Board geholt. Der Verifier wählt daraus eine Wahl, die er Auszählen möchte. Erst jetzt werden die Gesamtdaten der Wahl vom Bulletin-Board abgeholt. Diese beinhalten die Wahldefinitionen, welche in ein Election-Objekt konvertiert werden, und eine Liste aller Stimmzettel der Wahl, welche in eine Liste von Ballot-Objekten überführt werden. Stimmzettel, welche nicht in Ballot-Objekte konvertiert werden können, werden als ungültig markiert.

Nun beginnt der Verifizierungssprozess: In einem ersten Schritt werden die Zeitstempel der Ballots mit der Wahlperiode im Election-Objekt verglichen. Es werden in diesem Schritt alle Ballots als ungültig erklärt, welche zu früh oder zu spät eingetroffen sind. Im nächsten Schritt werden die NIZKPs  $\pi_1$ ,  $\pi_2$  und  $\pi_3$  verifiziert. Für die Verifizierung der Beweise müssen die Proofsystem-Objekte, welche auch für

die Erstellung der individuellen Beweise benutzt wurden, wieder hergeleitet werden. Da die Proofsheets selbst von der jeweiligen Stimme  $e$  abhängig sind, muss für jede einzigartige Stimme jeweils eine eigene Gruppe von Proofsheets erstellt werden. Mit diesen werden anschliessend die Beweise verifiziert und alle Ballots, deren Beweise ungültig sind, werden entsprechend markiert. Im letzten Schritt wird nun unter allen gültigen Ballots nach Mehrfachstimmen des gleichen Wählers gesucht. Diese Mehrfachstimmen können dadurch erkannt werden, dass sie denselben Wert für das ElectionCredential  $\hat{u}$  besitzen. Werden Mehrfachstimmen entdeckt, wird die Stimme, welche als erstes auf dem Bulletin-Board eingetroffen ist, ausgewählt und alle übrigen Stimmen desselben Wählers werden als ungültig erklärt.

Die übrig gebliebenen Stimmen werden nun ausgezählt. Dabei wird kontrolliert, ob die Stimmen die Definition der Wahl erfüllen. Ist dies der Fall so werden die Stimmen zum Wahlresultat addiert. Stimmen, welche die Wahldefinitionen nicht erfüllen werden verworfen und werden nicht in das Wahlresultat einbezogen. Ist dies der Fall, so wird darauf verzichtet einen weiteren Stimmzettel desselben Wählers zu suchen.

Das erhaltene Wahlresultat wird in einem ElectionResult-Objekt gespeichert und dem Verifier angezeigt. Ein ElectionResult besteht sowohl aus dem Resultat der Wahl sowie aus der Liste der validierten Ballots. Wahlweise kann nun das Resultat auf dem Bulletin-Board publiziert werden. Wird dies ausgewählt so wird das Wahlresultat in ein JSON-Objekt konvertiert und signiert an das Bulletin-Board übertragen.

## 5.7. Testing

Das Testing des Wahlsystems umfasst mehrere Aspekte. Zum einen müssen die Funktionen der individuellen Komponenten an sich getestet werden und zum anderen muss auch verifiziert werden, dass die Komponenten korrekt zusammenarbeiten. Für die Tests der individuellen Komponenten wurden Unit-Tests definiert. Um das Zusammenspiel der Komponenten zu überprüfen, wurden Systemtests erarbeitet.

Die Systemtests wurden mit Hilfe von Test Cases definiert. Diese bestehen aus einer Liste von Instruktionen, welche von einem Tester durchgeführt werden müssen. Basierend auf den Use Cases wird hier ein Tester aufgefordert, eine ganze Wahl von Anfang bis Ende durchzuspielen. Die einzelnen Test Cases wurden von den Use Cases abgeleitet und werden in der Reihenfolge „Wähler Registrierung“, „Wahl Definition“, „Stimmabgabe“ und „Auszählung – Verifikation“ durchgeführt. Dabei bauen die Test Cases aufeinander auf, so dass jeweils die Resultate aus dem vorangegangen Test Case als Eingabe für den nächsten verwendet werden. Konnten auf diese Weise alle Test Cases erfolgreich durchlaufen werden, gilt der Test als erfolgreich. Mithilfe dieser Test Cases wird auch zusätzlich die Benutzeroberfläche getestet. Die Definitionen der Test Cases sind im Anhang F des Dokuments einsehbar.

Mit den Unit Tests werden zusätzlich die Funktionen der einzelnen Programmteile getestet. Dabei wurden UnitTests für die Java-Applikationen mit JUnit definiert. Für die UnitTests des Bulletin-Board wurde PHPUnit verwendet. Bei den UnitTest wurde der Fokus vor allem auf die korrekte Funktionsweise der Verifizierungs- und Validierungsprozesse gelegt, da es in diesen Fällen entscheidend ist, falsche Beweise oder Signaturen korrekt als solche zu erkennen.

## 5.8. Integration von Tor

Im vorangehenden Kapitel 3.3 wurden bereits die theoretischen Aspekte rund um Tor und die Integration ins Protokoll betrachtet. Nachfolgend soll die effektive Implementation erläutert werden.

### 5.8.1. Umsetzung der Tor-Anbindung

**Tor-Anbindung integriert in VoterApp** Die eleganteste Möglichkeit eine Tor-Anbindung für die VoterApp zu realisieren, wäre die Integration des Tor-Proxys in die App. Dieser Ansatz hätte viele Vorteile, wie zum Beispiel die Ausführung der App im Benutzerkontext, die verbesserte Usability für den Endkunden und die bessere Wartbarkeit bedingt durch verringerte externe Abhängigkeiten. Der Nachteil dieser Lösung ist der erhöhte Implementationsaufwand und die höhere Fehleranfälligkeit, da der Proxy unter Umständen nicht auf demselben Protokollstand ist wie der Rest des Netzwerks.

**Tor-Anbindung über separaten Proxy-Dienst** Der einfachere Weg der Tor-Anbindung ist die Nutzung eines bereits vorhandenen lokalen Proxy-Dienstes. Dies hat den Nachteil, dass der Benutzer einen separaten Dienst installieren muss und höchstwahrscheinlich dazu erhöhte Rechte brauchen wird. Aus Usability-Sicht ist dieser Ansatz schlechter und wäre auch aus Wartbarkeits-Sicht nicht optimal. Allerdings sprechen Argumente wie der geringere Ressourcenaufwand und die erhöhte Sicherheit - da der Dienst aktiv von der Tor-Community gewartet wird - für diese Lösung. Der Tor-Dienst kann über die Kommandozeile ausgeführt werden und baut danach einen Tor-Circuit auf. Zu erwähnen ist hier, dass er *per default* erst nach rund 10 Minuten einen neuen Tor-Circuit aufbaut, unabhängig ob die VoterApp inzwischen neugestartet wurde.

**Umsetzungsentscheid** Da es nicht möglich war innert nützlicher Frist die Tor-Anbindung zu integrieren, wurde der vom Tor-Projekt entwickelte Tor-Dienst verwendet. Dies geschah in Absprache mit den Betreuern. Der Tor-Dienst steht über die Website des Projekts [8] im Downloadbereich als Source Code oder kompiliert für Linux und macOS zur Verfügung.

### 5.8.2. Implementation

Die Tor-Anbindung mithilfe eines lokalen Proxy-Dienstes war aufgrund des modularen Aufbaus gut möglich. Es wurden verschiedene Komponenten adaptiert oder hinzugefügt, welche nachfolgend aufgelistet und beschrieben sind.

**Tor Proxy-Service** Der Proxy-Service wurde über Homebrew<sup>2</sup> mit dem Befehl `brew install tor` auf einem MacBook Pro installiert. Die Standard-Konfiguration des Tor-Proxys richtet einen SOCKS-Proxy auf dem Port 9050 ein, welcher über die lokale Loopback-IP 127.0.0.1 zur Verfügung steht. Der Tor-Proxy wird über den Aufruf `tor` im macOS-Terminal gestartet und baut eine Verbindung zum Tor-Netzwerk auf.

**VoterApp** Die VoterApp musste nur marginal angepasst werden, weil aufgrund des Software-Designs im Java-Package `voterApp` nur App-spezifische Komponenten, wie beispielsweise die graphische Oberfläche, enthalten sind. Der Bildschirm zur Stimmabgabe wurde um eine Checkbox erweitert, welche es dem Nutzer erlaubt Tor zur anonymen Übermittlung der Stimme zu wählen. Da standardmäßig das Anonymisierungsnetzwerk verwendet werden soll, ist die Checkbox aktiviert. Auf dem darauffolgenden Übersichtsscreen, welcher die aktuelle Stimme nochmals anzeigt, wird dem Benutzer ebenfalls angezeigt, ob die Stimme über Tor gesendet wird.

---

<sup>2</sup>Homebrew ist ein Paketmanager für macOS, welcher verschiedene Tools zur Installation über die Kommandozeile anbietet. Weitere Infos auf der Website von Homebrew [3]

**Util-Package** Im Package *util* musste primär die Klasse *CommunicationController* angepasst werden. Die bestehende Funktion *postJsonStringToURL(String url, String json)* wurde ergänzt und erwartet nun auch den Boolean-Wert *Boolean useTor* beim Aufruf. Wenn dieser Boolean-Wert *false* ist, wird das Ballot wie bisher über einen HTTP POST-Aufruf direkt an das Bulletin-Board gesendet. Wenn er aber *true* ist, wird das Ballot über den lokalen SOCKS-Proxy und somit über das Tor-Netzwerk gesendet.

Standardmäßig unterstützt der SOCKS-Proxy-Connector des Apache-HTTP-Frameworks keine SSL/TLS-ge sicherten Verbindungen. Damit der Exit-Node des Tor-Netzwerks die Stimme nicht einsehen oder manipulieren kann, muss aber zwingend eine TLS-Verbindung über das Tor-Netzwerk hinaus bis zum Bulletin-Board bestehen. Die bestehende *SSLConnectionSocketFactory* musste deshalb erweitert werden.

**Bulletin-Board** Das Bulletin-Board wurde um die Funktion erweitert, dass es die IP-Adresse des Vot-ers zusammen mit dem Ballot, Timestamp, etc. in der Datenbank ablegt. Dies bietet die Möglichkeit Auswertungen zu machen, welche Stimmen über Tor abgegeben wurden und welche beispielsweise aus dem BFH-Netz (147.87.0.0/16) stammen.



## 6. Ausblick

Der realisierte Proof of Concept zeigt, dass der produktive Betrieb einer Infrastruktur mit bedingungslosem Wahlgeheimnis möglich ist. Allerdings wurden aufgrund des zeitlichen Rahmens und der Abgrenzungen einige Aspekte weggelassen, welche für eine produktive Version adressiert werden müssen. Nachfolgend werden einige dieser Aspekte aufgezeigt und Lösungsansätze beschrieben.

**„Append-only“ Bulletin-Board** Im Rahmen dieser Bachelor-Thesis war bereits die Verwendung eines Bulletin-Board mit „append-only“ Eigenschaft als mögliches Kann-Ziel integriert; dies wurde aber aus Zeitgründen nicht implementiert. Auch wenn das implementierte Bulletin-Board die benötigten Funktionen für den Prototyp des Wahlsystems bietet, ist die Vertrauensannahme in die Betreiber hoch. Severin Hauser, Mitglied der E-Voting-Gruppe der BFH, hat ein Bulletin-Board mit „append-only“ Eigenschaft implementiert. Sein Design erlaubt keine nachträglichen Mutationen von Wahldaten und Ballots, was nachvollziehbare Wahlen fördern würde.

**Integration Tor Proxy anstatt Verwendung Proxy-Service** Um den Prototyp in ein fertiges Produkt zu überführen, müsste der Tor Proxy in die VoterApp integriert werden. Auch wenn der externe Proxy aktiv von der Tor-Community gepflegt wird und somit den aktuellen Tor-Spezifikationen entspricht, ist ein externer Proxy-Service aus verschiedenen Gründen nachteilig. Beispielsweise bedeutet aus Entwicklersicht eine externe Komponente einen Mehraufwand bei Wartung und Testing, sowie eine mögliche zusätzliche Fehlerquelle. Exemplarisch könnte beispielsweise eine lokal installierte Firewall den Netzwerkverkehr zwischen VoterApp und Service sperren oder beeinträchtigen. Auch aus Benutzersicht bringt ein Proxy-Service Nachteile mit sich: Einerseits könnte der Endanwender durch einen weiteren (Hintergrund-) Dienst unnötig verwirrt werden und andererseits wurden in Vergangenheit solche lokalen Proxys oft durch Malware verwendet.

Die vorangegangene Argumentation zeigt einige Gründe, warum eine Integration des Tor Proxys in die VoterApp sinnvoll ist. Recherchen haben gezeigt, dass es bereits verschiedene Java-Frameworks für Tor gibt. „Orchid“ von Subgraph [5] oder die „Tor\_Onion\_Proxy\_Library“ des Thali Projekts [6] bieten beispielsweise open source Java-Implementationen eines Tor-Proxys.

**Verschlüsselung der Stimmen und Verteilung der Secret Keys** In der aktuellen Implementation werden Stimmen encodiert über einen verschlüsselten Kanal gesendet. Danach werden die Stimmen im Klartext auf dem Bulletin-Board abgelegt. Mithilfe von Public-Key-Kryptographie könnten die Stimme bereits in der VoterApp verschlüsselt und anschliessend auch verschlüsselt auf dem Bulletin-Board abgelegt werden. Diese Verschlüsselung würde die Fairness einer Wahl sicherstellen, da erst am Ende einer Wahl die Wahldaten entschlüsselt und damit das Wahlresultat berechnet werden kann.

Es stellt sich allerdings die Frage, wer nun diesen privaten Schlüssel zum Entschlüsseln der Wahl besitzt und wann er ihn benutzen darf. Wenn der private Schlüssel in den Händen einer einzigen Person oder Institution ist, erhält diese einen grossen Einfluss auf die Wahl. Da das Bulletin-Board öffentlich zugänglich ist, kann sie sich die Wahldaten herunterladen und entschlüsseln, wodurch sie jederzeit das aktuelle Wahlresultat kennt. Der Ansatz des Secret Sharings würde es erlauben, einen geheimen Schlüssel auf mehrere

Parteien zu verteilen und somit die Macht eines einzelnen zu mindern. Dies würde die Durchführung einer fairen Wahl mit schwachen Vertrauensannahmen erlauben und wäre somit anzustreben.

**Ausbau der Abstimmungsoptionen** Bei der heutigen Papierwahl sind verschiedenste Abstimmungsoptionen möglich, welche im aktuellen Prototypen von Anfang an nicht Teil der Arbeit waren. Beispielsweise ist es in der Schweiz üblich, bei Parlamentswahlen „zu panaschieren“, also verschiedene Wahllisten zu kombinieren oder eine Einzelne zu ergänzen. Es gibt teilweise auch Gegenvorschläge des Bundes zu Volksinitiativen, welche entsprechend dargestellt werden sollten. Das Protokoll nach Haenni und Locher [2] und auch der modulare Aufbau des Proof of Concept schliessen weitere Abstimmungsoptionen nicht aus. Diese könnten bei Bedarf implementiert werden.

## 7. Fazit

**Resultat** Das Resultat dieser Bachelor-Thesis ist ein funktionierender Prototyp eines elektronischen Wahlsystems. Der Proof of Concept zeigt die Machbarkeit einer modularen Software, welche dem Wähler ein bedingungsloses Wahlgeheimnis bietet. Die offene Implementation des Protokolls von Haenni und Locher erlaubte es, einen authentischen Kanal zu implementieren, indem Nachrichten jeweils signiert an das Bulletin-Board übertragen werden. Einzig die Stimmabgabe wird nicht signiert, da dies die Anonymität des Wählers aufheben würde. In einem weiteren Schritt wurde die Möglichkeit implementiert, dass der Wähler seine Stimme anonym über das Tor-Netzwerk abgeben kann. Diese Erweiterung ist neuartig und zeigt die Realisierbarkeit der Vorschläge von Haenni und Locher.

**Lessons learned** Die Arbeit an der Bachelor-Thesis war eine interessante Herausforderung für die Autoren am Ende Ihres Bachelor-Studiums. Der zeitliche Rahmen von 16 Wochen war leider zu eng, um sämtliche Kann-Ziele zu erreichen. Trotzdem muss an dieser Stelle erwähnt werden, dass mit der Implementation von Tor in den Prototypen ein wichtiges Kann-Ziel erreicht wurde. Vor dem Erarbeiten der Kann-Ziele mussten zuerst die beiden Muss-Ziele „Wahl durchführen“ und „Recherche anonymer Kanal“ erreicht werden. Auch wenn im Rahmen des Vorprojekts viele Grundlagen erarbeitet wurden, wurde die Einarbeitungszeit in die Technologien unterschätzt. Allerdings konnte der Verzug innerhalb weniger Wochen wieder gut gemacht werden, was nur dank gutem Teamwork und einigen Zusatzaufwänden funktionierte. An dieser Stelle muss aber ebenfalls erwähnt werden, dass ohne die im Vorprojekt selbstständig erarbeiteten Kenntnisse diese Arbeit nicht möglich gewesen wäre.

**Ausblick** Damit der Proof of Concept in eine produktives Wahlsystem überführt werden kann, müssen noch einige Themen adressiert werden. Diese wurden aufgrund des Projektrahmens weggelassen oder konnten aus zeitlichen Gründen nicht mehr betrachtet werden. Beispielsweise wird in der vorliegenden Version eine Implementation des Bulletin-Boards verwendet, welches nicht gegen nachträgliche Mutation an Wahldaten und Ballots geschützt ist. Um faire Wahlen zu gewährleisten, müssten die Stimmen zudem bei Abgabe verschlüsselt und erst nach Wahlende entschlüsselt werden können. Schliesslich müssten noch Fragen wie die Integration des Tor-Proxy-Services in die VoterApp, den Ausbau der Abstimmungsoptionen oder den Aufbau einer öffentlichen Zertifikatsverwaltung (PKI) beantwortet werden.

**Schlusswort** Die Erkenntnisse dieser Arbeit sind vielfältig. Erstens wurde gezeigt, dass das E-Voting Protokoll „*Verifiable Internet Elections with Everlasting Privacy and Minimal Trust*“ von Haenni und Locher in ein reales Wahlsystem überführt werden kann. Zweitens konnte gezeigt werden, dass es in der Tat möglich ist, Tor als anonymen Kanal zur Stimmabgabe zu verwenden und damit die bereits im Protokoll gebotene Privacy auszubauen. Schliesslich wurde eine mögliche Basis gelegt für weitere wissenschaftliche Arbeiten rund um das Thema der elektronischen Wahlen mit bedingungslosem Wahlgeheimnis.



# Selbständigkeitserklärung

Wir bestätigen, dass wir die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der im Literaturverzeichnis angegebenen Quellen und Hilfsmittel angefertigt haben. Sämtliche Textstellen, die nicht von uns stammen, sind als Zitate gekennzeichnet und mit dem genauen Hinweis auf ihre Herkunft versehen.

Ort, Datum: Bern, 19.01.2017

Namen Vornamen: Bürk Timo Nellen Sebastian

Unterschriften: .....



# Literaturverzeichnis

- [1] T. Bürk and S. Nellen, "Projekt 2: E-Voting. Abschlussbericht Modul 'BTI-7302 Projekt 2'," 2016.
- [2] R. Haenni and P. Locher, "Verifiable internet elections with everlasting privacy and minimal trust," 2015.
- [3] "Website von Homebrew, dem fehlenden Paketmanager für macOS," Homebrew Projekt, 2016, visited on 2017-01-11. [Online]. Available: [http://brew.sh/index\\_de.html](http://brew.sh/index_de.html)
- [4] "Wiki-Seite über Tor (griechisch)," skytal.es, 2014, visited on 2017-01-08. [Online]. Available: <https://skytal.es/wiki/Tor>
- [5] "Subgraph Webseite zu Orchid," subgraph, 2014, visited on 2017-01-08. [Online]. Available: <https://subgraph.com/orchid/download/index.en.html>
- [6] "Github-Repository der Tor\_Onion\_Proxy\_Library," Thali Project, 2016, visited on 2017-01-11. [Online]. Available: [https://github.com/thaliproject/Tor\\_Onion\\_Proxy\\_Library](https://github.com/thaliproject/Tor_Onion_Proxy_Library)
- [7] "Tor: Overview," The Tor Project, visited on 2017-01-14. [Online]. Available: <https://www.torproject.org/about/overview.html.en>
- [8] "Website des Tor Projekts," The Tor Project, visited on 2017-01-08. [Online]. Available: <https://www.torproject.org>
- [9] "MVC Framework Course," Webtech Learning, visited on 2017-01-16. [Online]. Available: <http://www.webtechlearning.com/mvc-framework-training-in-chandigarh/>
- [10] "Arbeitsweise von Tor," Wikimedia Commons, 2013, visited on 2017-01-08. [Online]. Available: [https://commons.wikimedia.org/wiki/File:TOR\\_Arbeitsweise.svg](https://commons.wikimedia.org/wiki/File:TOR_Arbeitsweise.svg)



# Abbildungsverzeichnis

2.1. Screenshot der Ziele des Teil 1, Durchführen einer Wahl . . . . .	3
3.1. Arbeitsweise des Tor Netzwerks aus [10] . . . . .	10
3.2. Aufbau des Onion Routing aus [4] . . . . .	11
4.1. Komponenten abcVote schematisch dargestellt . . . . .	14
4.2. Kommunikation innerhalb abcVote . . . . .	15
5.1. REST-Schnittstellen des Bulletin-Boards mit entsprechenden HTTP-Befehlen (GET, POST) . . . . .	18
5.2. ERM-Diagramm der abcVote Datenbank . . . . .	19
5.3. MVC Design-Pattern aus [9] . . . . .	22



## **A. Bachelorthesis-Aufgabe**



## Bachelorthesis-Aufgabe

für Timo Bürk  
Sebastian Nellen  
Abteilung Informatik  
Betreuung durch Prof. Dr. Rolf Haenni  
Prof. Philipp Locher

### **Elektronische Wahlen mit bedingungslosem Wahlgeheimnis**

Bei elektronischen Wahlen ist das Garantieren des Stimmgeheimnisses ein wesentlich schwierigeres Problem als bei klassischen Wahlen mit Papierstimmen. Es ist mittels kryptographischen Methoden zwar möglich, das Mischen der Stimmzettel elektronisch nachzubilden, die Sicherheit dabei ist aber immer nur so stark wie die verwendeten kryptographischen Verfahren und Parameter. Obwohl diese heute als sicher gelten, kann man davon auszugehen, dass in Zukunft, zum Beispiel in 30, 50 oder 100 Jahren, diese nicht mehr genügen. Somit besteht ein Risiko, dass das Stimmgeheimnis nachträglich noch gebrochen werden kann.

In der Literatur der kryptographischen Wahlprotokolle gibt es verschiedene Vorschläge für Systeme, in denen das Wahlgeheimnis bedingungslos geschützt werden kann, d.h. unabhängig des technischen Fortschritts und von neuen wissenschaftlichen Erkenntnissen. In diesem Projekt geht es darum, eines dieser Systeme zu analysieren und in Form einer Prototypen zu implementieren.

Beginn der Arbeit 19. September 2016  
Abschluss der Arbeit 19. Januar 2017

Der Betreuer:

Der Abteilungsleiter:

## B. Use Cases



## Anwendungsfallbeschreibung (Use Case Scenario)

Nr. und Name:	1: Wähler Registrierung
Szenario:	Der Wähler registriert sich auf der elektronischen Abstimmungsplattform.
Kurzbeschreibung:	Die Wähleridentität des Wählers wird an das Bulletin-Board gesendet.
Beteiligte Akteure:	Wähler
Auslöser / Vorbedingung:	Der Wähler möchte an einer kommenden Wahl elektronisch abstimmen. / Wähler verfügt über ein digitales Zertifikat und einen passenden privaten Schlüssel. Der Wähler resp. sein Gerät verfügt über eine funktionierende Verbindung ins Internet.
Ergebnisse / Nachbedingung:	Das Bulletin-Board kennt das authentisierte/signierte Wähler-Credential des Wählers.
Bemerkungen:	Die verwendeten Werte (z.B. $h_1, h_2$ ), die Begrifflichkeiten (z.B. Wähler-Credential) und das Protokoll orientieren sich am Paper „Verifiable Internet Elections with Everlasting Privacy and Minimal Trust“ von Locher und Haenni.

### Ablauf:

Nr.	Wer	Was
1.01	Wähler	Der Wähler öffnet die Voter-App über die ausführbare Datei oder eine Verknüpfung.
1.02	Voter-App	Die Voter-App zeigt dem Wähler den Hauptbildschirm an.
1.03	Wähler	Der Wähler betätigt den Button „Wähler Registrierung“ um den Registrierungsprozess zu starten.
1.04	Voter-App	Die Voter-App fragt beim Bulletin-Board die Werte $h_1, h_2, p$ und $q$ an, um damit das Wähler-Credential $u$ zu berechnen.
1.05	Bulletin-Board	Das Bulletin-Board liefert die Werte $h_1, h_2, p$ und $q$ .
1.06	Voter-App	Die Voter-App informiert den Wähler, dass sie nun sein öffentliches und privates Wähler-Credential generieren wird.
1.07	Voter-App	Die Voter-App generiert für den Wähler ein öffentliches und privates Wähler-Credential, speichert diese in den lokalen Keystore ab und zeigt diese dem Wähler in einer geeigneten Form an.
1.08	Wähler	Der Wähler klickt nun auf den Button „Wähler-Credential übertragen“.
1.09	Voter-App	Die Voter-App zeigt den Dialog „Wähler-Credential signieren“ an.
1.10	Wähler	Der Wähler wählt den privaten Schlüssel aus dem lokalen Keystore zum Signieren seines Wähler-Credentials aus und stößt durch Klick auf den Button „Signieren“ den Signaturvorgang an.
1.11	Voter-App	Die Voter-App signiert die öffentliche Wähleridentität mit dem angegebenen privaten Schlüssel.
1.12	Voter-App	Die Voter-App überträgt das signierte öffentliche Wähler-Credential des Nutzers an das Bulletin-Board.
1.13	Bulletin-Board	Das Bulletin-Board nimmt die öffentliche Wähleridentität des Nutzers entgegen und überprüft die Signatur. Es quittiert den Erhalt und das Ergebnis der Signaturüberprüfung an die Voter-App.



1.14	Voter-App	Die Voter-App informiert den Benutzer über den erfolgreichen Abschluss des Vorgangs.
------	-----------	--

**Ausnahmen, Varianten:**

Nr.	Wer	Was
1.12.1	Voter-App	Sollte der Signierungs-Prozess nicht erfolgreich abgeschlossen werden können, wird der Benutzer darüber informiert.
1.14.1	Voter-App	Sollte die Signaturüberprüfung fehlschlagen, wird der Benutzer darüber informiert.



## Anwendungsfallbeschreibung (Use Case Scenario)

Nr. und Name:	2: Wahl Definition
Szenario:	Die Wahladministration definiert eine neue Wahl.
Kurzbeschreibung:	Die Wahladministration erstellt eine neue Wahl und definiert die Rahmenbedingungen sowie die zugelassenen Wähler.
Beteiligte Akteure:	Wahladministration
Auslöser / Vorbedingung:	Es soll eine Wahl durchgeführt werden. / Eine Mindestanzahl Wähleridentitäten muss vorhanden sein.
Ergebnisse / Nachbedingung:	Die Wahl ist publiziert und kann durchgeführt werden. Die stimmberechtigten Wähler wurden informiert.
Bemerkungen:	Die verwendeten Werte (z.B. $h_1, h_2$ ), die Begrifflichkeiten (z.B. Wähler-Credential) und das Protokoll orientieren sich am Paper „Verifiable Internet Elections with Everlasting Privacy and Minimal Trust“ von Locher und Haenni.

### Ablauf:

Nr.	Wer	Was
2.01	Wahladmin	Der Wahladministrator (Wahladmin) öffnet das Admin-App über die Ausführbare Datei oder eine Verknüpfung.
2.02	Wahladmin	Der Wahladministrator navigiert zum Bereich zur Erstellung einer neuen Wahl.
2.03	Admin-App	Verlangt eine Liste aller registrierten Wähler vom Bulletin-Board.
2.04	Bulletin-Board	Das Bulletin-Board sendet die Liste aller registrierter Wähler an das Admin-App.
2.05	Admin-App	Erhält die Liste der registrierten Wähler und stellt diese für den Benutzer dar.
2.06	Wahladmin	Der Wahladministrator wählt aus der angezeigten Liste aller registrierten Wähler alle stimmberechtigten Wähler für die neue Wahl aus.
2.07	Wahladmin	Der Wahladministrator erstellt die Optionen, welche in dieser Wahl ausgewählt werden können. Dabei können beliebig viele Optionen angegeben werden und mit einer kurzen Beschreibung versehen werden.
2.08	Wahladmin	Nachdem erstellen der Optionen gibt der Wahladministrator an, wie viele Optionen aus der Auswahl ausgewählt werden können.
2.09	Wahladmin	Der Wahladministrator definiert die Periode in welcher die Wähler ihre Stimmen abgeben können mit einem Start- und Enddatum.
2.10	Wahladmin	Nach der Eingabe aller Daten „schliesst“ der Wahladministrator die Wahl ab.
2.11	Admin-App	Das Admin-App stellt alle eingegebenen Daten zur neuen Wahl nochmals in einer Übersicht dar.
2.12	Wahladmin	Der Wahladministrator überprüft seine Eingaben und gibt anschliessend den Befehl zum Versenden der neuen Wahl.
2.13	Admin-App	Das Admin-App berechnet die Koeffizienten A des Polynoms P aus der Liste wahlberechtigten Wählern U und bestimmt den Abstimmungsgenerator $\hat{h}$ .
2.14	Admin-App	Das Admin-App ruft seine eigene Versionsnummer ab, um sie gemeinsam mit der Wahldefinition zu verschicken.
2.15	Admin-App	Das Admin-App zeigt den Dialog „Wahldefinition signieren“ an.



2.16	Admin-App	Der Benutzer wird aufgefordert seinen privaten Schlüssel für die Signatur aus dem lokalen Keystore auszuwählen.
2.17	Wahladmin	Der Wahladmin wählt seinen privaten Schlüssel zu seinem digitalen Zertifikat zum Signieren der Wahldefinition aus und stößt durch Klick auf den Button „Signieren“ den Signaturvorgang an.
2.18	Admin-App	Das Admin-App signiert die Wahldefinition mit dem angegeben privaten Key.
2.19	Admin-App	Das Admin-App sendet die Eingaben des Wahladministrators für die neue Wahl an das Bulletin-Board.
2.20	Bulletin-Board	Das Bulletin-Board überprüft die Signatur der erhaltenen Daten. Ist die Signatur korrekt, so werden alle Informationen der Wahl in die Datenbank des Bulletin-Boards abgelegt. Es quittiert die Speicherung und das Ergebnis der Signaturüberprüfung an das Admin-App.
2.21	Admin-App	Das Admin-App informiert den Benutzer über den erfolgreichen Abschluss des Vorgangs.

**Ausnahmen, Varianten:**

Nr.	Wer	Was
2.14.1	Admin-App	Das Admin-App stellt Fehler in den Eingaben fest. Es zeigt diese dem Wahladministrator an.
2.20.2	Admin-App	Das Admin-App informiert den Benutzer, über die ungültigen Angaben und zeigt diesem denselben Bildschirm wie bei 2.11 an.
2.18.1	Admin-App	Sollte der Signierungs-Prozess nicht erfolgreich abgeschlossen werden können, wird der Benutzer darüber informiert.
2.20.1	Bulletin-Board	Die Validierung der Signatur ist fehlgeschlagen. Das Bulletin-Board informiert das Admin-App über den bemerkten Fehler.



## Anwendungsfallbeschreibung (Use Case Scenario)

Nr. und Name:	3: Stimmabgabe
Szenario:	Der Wähler gibt seine Stimme ab.
Kurzbeschreibung:	Die Stimme („Ballot“) des Wählers wird generiert und an das Bulletin-Board gesendet.
Beteiligte Akteure:	Wähler
Auslöser / Vorbedingung:	Der Wähler möchte seine Stimme abgeben. / Der Wähler ist registriert und für diese Wahl stimmberechtigt.
Ergebnisse / Nachbedingung:	Die Stimme des Wählers ist auf dem Bulletin-Board eingetragen.
Bemerkungen:	Die verwendeten Werte (z.B. $h_1, h_2$ ), die Begrifflichkeiten (z.B. Wähler-Credential) und das Protokoll orientieren sich am Paper „Verifiable Internet Elections with Everlasting Privacy and Minimal Trust“ von Locher und Haenni.

### Ablauf:

Nr.	Wer	Was
3.01	Wähler	Der Wähler öffnet die Voter-App über die Ausführbare Datei oder eine Verknüpfung.
3.02	Voter-App	Die Voter-App zeigt dem Wähler den Hauptbildschirm an.
3.03	Wähler	Der Wähler betätigt den Button „Stimme abgeben“ um den Wahlprozess zu starten.
3.04	Voter-App	Die Voter-App schickt eine Anfrage an das Bulletin-Board und verlangt alle laufenden Abstimmungen.
3.05	Bulletin-Board	Das Bulletin-Board bestimmt aufgrund des aktuellen Datums alle offenen Abstimmungen und schickt die Kopfdaten (ID, Titel, Wahlperiode) der Abstimmungen an die Voter-App.
3.06	Voter-App	Die Voter-App stellt die Liste aller laufenden Daten für den Benutzer dar.
3.07	Wähler	Der Wähler selektiert eine Wahl und drückt den Button „Stimme abgeben“.
3.08	Voter-App	Die Voter-App liest die ID der selektierten Wahl aus und schickt eine Anfrage an das Bulletin Board und verlangt alle Daten zu dieser Wahl.
3.09	Bulletin-Board	Das Bulletin-Board schickt die Wahldefinitionen dieser Wahl an die Voter-App.
3.10	Voter-App	Die Voter-App nimmt die Wahldefinitionen entgegen und zeigt die Wahloptionen für den Benutzer an.
3.11	Wähler	Der Wähler wählt aus den Optionen $k$ aus (entsprechend der Wahldefinition) und bestätigt seine Wahl.
3.12	Voter-App	Die Voter-App überprüft die ausgewählten Optionen.
3.13	Voter-App	Die Voter-App fordert den Benutzer auf seine Private Credentials aus dem lokalen Keystore auszuwählen.
3.14	Wähler	Der Benutzer wählt seine Private Credentials aus.
3.15	Voter-App	Die Voter-App prüft, ob die angegebenen Private Credentials für diese Wahl zugelassen sind indem sie das Public Credential u berechnet und in das Polynom mit den Koeffizienten $A$ einsetzt.



3.16	Voter-App	Die Voter-App nimmt die ausgewählten Optionen, encodiert diese und formt damit die Stimme e.
3.17	Voter-App	Die Voter-App berechnet aus den Private Credentials, der Stimme e, den Koeffizienten A und dem Abstimmungsgenerator $\hat{h}$ das Ballot B = (c, d, e, $\hat{u}, \pi_1, \pi_2, \pi_3$ ).
3.18	Voter-App	Voter-App schickt das Ballot B an das Bulletin-Board.
3.19	Bulletin-Board	Das Bulletin-Board nimmt die Stimme entgegen und legt diese in der Datenbank ab.
3.20	Bulletin-Board	Das Bulletin-Board quittiert der Voter-App, dass es die Stimme erhalten hat.
3.21	Voter-App	Die Voter-App bestätigt dem Wähler, dass die Stimme erfolgreich übertragen wurde.

#### Ausnahmen, Varianten:

Nr.	Wer	Was
3.06.1	Bulletin-Board	(Optional) Mit einem Button kann der Benutzer die Liste filtern lassen, so dass nur noch Wahlen angezeigt werden, bei welchen der Benutzer stimmberechtigt ist.
3.06.2	Voter-App	Damit die Voter-App weiß, für welche Wahlen der Wähler berechtigt ist, benötigt sie die Liste der offenen Wahlen mit den zugehörigen Koeffizienten A ab vom Bulletin-Board.
3.06.3	Bulletin-Board	Das Bulletin-Board schickt der Voter-App eine Liste mit den offenen Wahlen und zugehörigen Koeffizienten A.
3.06.4	Voter-App	Wählt der Benutzer die Möglichkeit zum Filtern der Liste, so wird er aufgefordert seine Private Credentials aus dem lokalen Keystore auszuwählen.
3.06.5	Wähler	Der Benutzer wählt seine Private Credentials aus.
3.06.6	Voter-App	Mit Hilfe der Private Credentials rechnet die Voter-App das Public Credential u aus und prüft für jede Abstimmung durch einsetzen von u in das Polynom mit den Koeffizienten A, ob der Wähler für die Abstimmung wahlberechtigt ist. Die Voter-App bereinigt die Abstimmungsdaten, um jene Wahlen, für welche der Wähler nicht zugelassen ist.
3.11.1	Voter-App	Wenn die ausgewählten Optionen nicht der Wahldefinition entsprechen oder ein anderer Fehler auftritt, wird der Benutzer informiert und aufgefordert, seine Auswahl zu korrigieren.



## Anwendungsfallbeschreibung (Use Case Scenario)

Nr. und Name:	4: Auszählung – Verifikation
Szenario:	Der Verifier / Auszählung verifiziert alle Stimmen und berechnet das Ergebnis.
Kurzbeschreibung:	Die Wahl wird nach Abschluss durch den Verifier / Auszählung überprüft und ausgewertet.
Beteiligte Akteure:	Verifier / Auszählung
Auslöser / Vorbedingung:	Die Wahl ist abgeschlossen und das Resultat muss berechnet werden. / Die Wahlfrist ist abgelaufen.
Ergebnisse / Nachbedingung:	Das Wahlresultat wurde ausgegeben.
Bemerkungen:	Die verwendeten Werte (z.B. $h_1, h_2$ ), die Begrifflichkeiten (z.B. Wähler-Credential) und das Protokoll orientieren sich am Paper „Verifiable Internet Elections with Everlasting Privacy and Minimal Trust“ von Locher und Haenni.

### Ablauf:

Nr.	Wer	Was
4.01	Verifier	Der Verifier (Auszählender) öffnet das Verifier-App über die ausführbare Datei oder eine Verknüpfung.
4.02	Verifier-App	Das Verifier-App erfragt die abgeschlossenen Wahlinformationen (Wahl-ID, Titel, Wahlperiode) vom Bulletin-Board.
4.03	Bulletin-Board	Das Bulletin-Board liefert die angefragten Informationen zu den abgeschlossenen Wahlen zurück.
4.04	Verifier-App	Das Verifier-App zeigt dem Verifier eine Übersicht der abgeschlossenen Wahlen an.
4.05	Verifier	Der Verifier wählt die zu überprüfende Wahl aus der ihm angezeigten Liste aus.
4.06	Verifier-App	Das Verifier-App fordert beim Bulletin-Board alle Ballots und Wahlinformationen (Koeffizienten $A_{B-Board}$ , Wählerliste $U$ ) der entsprechenden Wahl an.
4.07	Bulletin-Board	Das Bulletin-Board liefert alle Ballots und Wahlinformationen der angefragten Wahl ab.
4.08	Verifier-App	Das Verifier-App berechnet mit der Wählerliste $U$ die Koeffizienten $A_{V-Tool}$ und vergleicht diese mit den Koeffizienten $A_{B-Board}$ (welche ursprünglich vom Wahldaten berechnet wurden und dann auf dem Bulletin-Board abgespeichert wurden).
4.09	Verifier-App	Das Verifier-App bestimmt in einer ersten Phase alle potenziell gültigen Ballots, in dem es die Time-Stamp des Stimmeingangs mit der zeitlichen Begrenzung der Wahl (Wahlperiode) vergleicht. Alle Ballots mit ungültigen Time-Stampen werden als ungültig markiert.
4.10	Verifier-App	Das Verifier-App bestimmt in einer zweiten Phase alle potenziell gültigen Ballots, in dem es die Beweise innerhalb der verbliebenen Ballots überprüft. Alle Ballots mit falschen Beweisen werden als ungültig markiert.
4.11	Verifier-App	Das Verifier-App bestimmt Mehrfachstimmen, in dem es nach jenen Ballots sucht, welche ein identisches Wähler-Credential $\hat{u}$ besitzen. Es wird jeweils die erste abgegebene Stimme verwendet und jede weitere als ungültig markiert.

4.12	Verifier-App	Das Verifier-App zählt nun die ausgewählte Abstimmung aus.
4.13	Verifier-App	Das Verifier-App zeigt dem Verifier das Abstimmungsresultat an. Neben dem Schlussresultat, sollten Zwischenschritte und weitere Informationen in geeigneter Form dargestellt werden. (Beispielsweise könnte angezeigt werden, dass von total X Ballots nur Y Ballots rechtzeitig eingehen und dass davon nur Z Ballots korrekte Beweise enthalten.)
4.14	Verifier	Der Verifier prüft die angezeigten Angaben und bestätigt diese per Klick auf den Button „Auszählung bestätigen“.
4.15	Verifier-App	Das Verifier-App fragt den Verifier über einen Dialog, ob er die ausgezählte Wahl signieren und ans Bulletin-Board übermitteln will.
4.16	Verifier	Der Verifier stimmt der Signierung und Übermittlung zu.
4.17	Verifier-App	Das Verifier-App zeigt den Dialog „Wahlresultat signieren“ an.
4.18	Verifier	Der Verifier wählt den privaten Schlüssel zum Signieren des ausgezählten Wahlresultats aus und stößt durch Klick auf den Button „Signieren“ den Signaturvorgang an.
4.19	Verifier-App	Das Verifier-App signiert das ausgezählte Wahlresultat mit dem angegeben privaten Schlüssel.
4.20	Verifier-App	Das Verifier-App überträgt das signierte Wahlresultat an das Bulletin-Board.
4.21	Bulletin-Board	Das Bulletin-Board nimmt das signierte Wahlresultat des Verifiers entgegen, überprüft die Signatur. Es quittiert den Erhalt und das Ergebnis der Signaturüberprüfung an das Verifier-App.
4.22	Verifier-App	Das Verifier-App informiert den Verifier über den erfolgreichen Abschluss des Vorgangs.

**Ausnahmen, Varianten:**

Nr.	Wer	Was
4.04.1	Verifier-App	(Optional) Sofern das Verifier-App Kenntnis hat, welche Wahlen es bereits verifiziert und ausgezählt wurden, können diese in der Übersicht entsprechend markiert werden.
4.08.1	Verifier-App	Die Koeffizienten <i>AV-Tool</i> (berechnet durch Verifier-App) und <i>Ab-Board</i> (auf dem Bulletin-Board hinterlegt, vom Wahldaten berechnet) sind unterschiedlich. Die Wahl ist somit ungültig.
4.16.1	Verifier	Der Verifier will die Auszählung nicht signieren und übermitteln. Er bricht deshalb den Vorgang ab in dem er auf die entsprechende Schaltfläche klickt.
4.16.2	Verifier-App	Das Verifier-App bricht den Signierungs-und-Übermittlungs-Vorgang ab und zeigt erneut den Startbildschirm (welcher alle auszählbaren Wahlen auflistet) an.
4.19.1	Verifier-App	Sollte der Signierungs-Prozess nicht erfolgreich abgeschlossen werden können, wird der Benutzer darüber informiert.
4.22.1	Verifier-App	Sollte die Signaturüberprüfung fehlschlagen, wird der Verifier darüber informiert.



## C. Anforderungen an Komponenten

**AdminApp** Die AdminApp bildet in Zusammenarbeit mit dem Bulletin-Board den Use Case „Wahl Definition“ ab und ermöglicht einem Benutzer eine neue Wahl zu definieren und zu publizieren.

- Lokale Userdaten
  - Keystore für digitales Signierungszertifikat mit passendem Private Key
- Wahl Definition
  - Eine Liste aller registrierten Wähler vom Bulletin-Board holen und anzeigen
  - Es muss eine Auswahl der berechtigten Wähler getroffen werden können
  - Die Wahloptionen und die Wahlperiode müssen definiert werden können
  - Die Wahlidentität  $\hat{h}$  muss bestimmt werden
  - Die fertige Wahldefinition muss signiert werden
  - Die signierte Wahldefinition wird an das Bulletin-Board übertragen

**VoterApp** Die VoterApp deckt die beiden Use Cases „Wähler Registrierung“ und „Stimmabgabe“ ab.

- Lokale Userdaten
  - Keystore
    - \* Ablage der Private Credentials des Wählers
    - \* Ablage des digitalen Signierungszertifikats mit passendem Private Key
- Wähler Registrierung
  - Erstellen von neuen Private Credentials
  - Berechnen des Public Credential aus Private Credentials
  - Signieren des berechneten Public Credential
  - Übermittlung des signierten Public Credential an das Bulletin-Board
- Abstimmung
  - Kopfdaten von allen laufenden Abstimmungen vom Bulletin-Board beziehen und anzeigen
  - Wahldefinition einer ausgewählten Wahl vom Bulletin-Board beziehen und anzeigen
  - Der Wähler muss aus den Wahloptionen die gewünschte Option auswählen können
  - Berechnen des Stimmzettels und Übermittlung dessen an das Bulletin-Board

**VerifierApp** Die VerifierApp wird für das Auszählen einer Wahl verwendet und ist somit für den Use Case „Auszählung-Verifikation“ zuständig.

- Auszählung-Verifikation
  - Laden einer Liste von Kopfdaten aller abgeschlossenen Wahlen vom Bulletin-Board
  - Laden sämtlicher Informationen zu einer bestimmten Wahl
  - Verifizieren und Berechnen des Wahlresultats
  - Signieren und Übermitteln des ausgezählten Wahlresultats an das Bulletin -Board

**Bulletin-Board** Das Bulletin-Board dient als zentrale Datenablage für das Wahlsystem und ist somit das Verbindungsstück zwischen den drei übrigen Komponenten. Seine Hauptaufgabe besteht dabei aus dem Ablegen und der Wiedergabe von Daten. Aufgrund der Erkenntnisse aus den Use Cases muss das Bulletin-Board folgende Anforderungen erfüllen:

- Einen neu registrierten Wähler in die Liste aller berechtigten Wählern aufnehmen
- Eine Liste aller vorhandenen Wählern wiedergeben
- Eine Definition für eine neue Wahl entgegennehmen und ablegen
- Eine Liste mit den Kopfdaten aller Wahlen wiedergeben
- Eine Definition einer bestimmten Wahl wiedergeben
- Ein Stimmzettel zu einer bestimmten Wahl entgegennehmen und ablegen
- Ein Wahlresultat zu einer bestimmten Wahl entgegennehmen und ablegen
- Publikation und/oder Wiedergabe von Wahlresultaten

## **D. Setup Guide**



## A setup guide for abcVote

### 1 Bulletin-Board

**Server-Environment:** Debian 8

*A more detailed guide (in german) is attached at the end of this document*

**To Do:**

- Install LAMP-Stack (Apache, MySQL, PHP) as described in <https://wiki.debian.org/LaMp>
- Activate mod\_rewrite for correct function of .htaccess
  - Edit /etc/apache2/apache2.conf
  - <Directory /var/www/>
    - Options Indexes FollowSymLinks
    - AllowOverride All
    - Require all granted
  - </Directory>
- Setup SSL-certificate with Let's encrypt! (Maybe you should rewrite http to https...)
- Install GIT client (git-core)
- Install Composer as described in <https://getcomposer.org/download/>
- Adopt apache-site to your needs
- Get Files from GIT <https://github.com/EVGStudents/abcVote>,  
copy the Bulletin-Board src files to e.g. /var/www/ and rename ./public/ to ./html/
- Run composer to install dependecies (e.g. "composer install" from within the /var/www/ directory)
- Don't forget to check ownership and access rights of the website
- Run the SQL scripts to create DB and add sample data

### 2 Java Applications

**Demo-Environment:** Apple OS X 10.11 El Capitan

**Needed Software:**

- Oracle Java 8 JDK and JRE
- Netbeans 8.2

**To Do:**

- Installation Homebrew according to <http://brew.sh>
- After Homebrew is installed, enter the following Terminal commands
  - brew install tor
- Get the source code of abcVote at <https://github.com/EVGStudents/abcVote>
- Copy the KeyStores into the Documents folder of your user's home (e.g. ~/Documents/abcVote)
- Add the projects to Netbeans
- Build the util-package
- Set the correct Bulletin-Board URL in each application's MainController
- Build the applications
- Run the applications
- You've to set the e-mail address corresponding to the certificate in the KeyStore file

Note: If you want to setup your own KeyStores and certificates you have to change the hard coded passwords in the apps or the set the KeyStore and certificate password to " Bern2016 "



## 3 Detailed German setup guide

### 3.1 Allgemein

Dieser Guide beschreibt die Einrichtung einer BFH-TI VM für die Verwendung als abcVote-Bulletin-Board

#### 3.1.1 SSH-Login auf Server

Auf Server einloggen per SSH und anschliessend via `su root` administrative Rechte erlangen.

#### 3.1.2 Updates installieren

```
root@nells-deb8:/home/bfh# apt-get update
root@nells-deb8:/home/bfh# apt-get upgrade
...
Do you want to continue? [Y/n] y
```

## 3.2 Benutzer

### 3.2.1 Benutzer erstellen

```
root@nells-deb8:/home/bfh# adduser nells1
Adding user `nells1' ...
Adding new group `nells1' (1002) ...
Adding new user `nells1' (1002) with group `nells1' ...
Creating home directory `/home/nells1' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for nells1
Enter the new value, or press ENTER for the default
  Full Name []: Sebastian Nellen
  Room Number []:
  Work Phone []: +41 79 737 57 52
  Home Phone []:
  Other []: nells1@bfh.ch
Is the information correct? [Y/n] y

root@nells-deb8:/home/bfh# adduser burkt4
Adding user `burkt4' ...
Adding new group `burkt4' (1003) ...
Adding new user `burkt4' (1003) with group `burkt4' ...
Creating home directory `/home/burkt4' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for burkt4
Enter the new value, or press ENTER for the default
  Full Name []: Timo Buerk
  Room Number []:
```



```
Work Phone []: +41 78 724 15 59
Home Phone []:
Other []: burkt4@bfh.ch
Is the information correct? [Y/n] y
```

### 3.2.2 Benutzer administrative Rechte geben

Basierend auf <https://wiki.debian.org/sudo>

```
root@nells-deb8:/home# adduser nells1 sudo
Adding user `nells1' to group `sudo' ...
Adding user nells1 to group sudo
Done.
root@nells-deb8:/home# adduser burkt4 sudo
Adding user `burkt4' to group `sudo' ...
Adding user burkt4 to group sudo
Done.
```

## 3.3 LAMP-Stack Installation

Basierend auf <https://wiki.debian.org/LaMp>

### 3.3.1 MySQL

```
root@nells-deb8:/home# apt-get install mysql-server mysql-client
```

→ Während Installation das Passwort des root-Benutzers festlegen.

### 3.3.2 Apache

```
root@nells-deb8:/home# apt-get install apache2 apache2-doc
```

### 3.3.3 PHP & phpMyAdmin

```
root@nells-deb8:/home/nells1# apt-get install php5 php5-mysql libapache2-mod-php5
root@nells-deb8:/home/nells1# apt-get install phpmyadmin
```

→ Während Installation „apache2“ als Webserver auswählen, sowie das Passwort des MySQL-Root-Benutzers eingeben. Das phpMyAdmin-Passwort habe ich vom System generieren lassen.

## 3.4 Weitere Software

### 3.4.1 GIT

```
root@nells-deb8:/home/nells1# apt-get install git-core
```



### 3.4.2 Webmin

```
root@nells-deb8:/home/nells1# cd /tmp
root@nells-deb8:/tmp# mkdir webmin
root@nells-deb8:/tmp# cd webmin
root@nells-deb8:/tmp/webmin# wget
http://prdownloads.sourceforge.net/webadmin/webmin_1.820_all.deb
root@nells-deb8:/tmp/webmin# apt-get install perl libnet-ssleay-perl openssl
libauthen-pam-perl libpam-runtime libio-pty-perl apt-show-versions python
libapt-pkg-perl
root@nells-deb8:/tmp/webmin# dpkg --install webmin_1.820_all.deb
```

Zugriff auf Webmin via [https://\[Server\]:10000](https://[Server]:10000)

## 3.5 Konfiguration Webserver

### 3.5.1 Aktivieren mod\_rewrite

```
root@nells-deb8:/home/nells1# a2enmod rewrite
root@nells-deb8:/home/nells1# nano /etc/apache2/apache2.conf
```

Damit anschliessend das .htaccess-File richtig interpretiert wird, muss im File </etc/apache2/apache2.conf> Folgender Eintrag wie folgt angepasst werden

```
<Directory /var/www/>
    Options Indexes FollowSymLinks
    AllowOverride All
    Require all granted
</Directory>
```

Neustart von Apache

```
root@nells-deb8:/home/nells1# service apache2 restart
```

### 3.5.2 Composer installieren

```
root@nells-deb8:/home/nells1# cd /var/www/
```

Bei einer späteren Neuinstallation sollte die Anleitung unter <https://getcomposer.org/download/> beachtet werden, damit die korrekten SHA384-Hashes zur Integritätsprüfung verwendet werden.

Nun folgt die oben erwähnte Anleitung (Stand 15.11.2016)

```
root@nells-deb8:/var/www# php -r "copy('https://getcomposer.org/installer',
'composer-setup.php');"
root@nells-deb8:/var/www# php -r "if (hash_file('SHA384', 'composer-
setup.php') ===
'aa96f26c2b67226a324c27919f1eb05f21c248b987e6195cad9690d5c1ff713d53020a02ac8c2
17dbf90a7eacc9d141d') { echo 'Installer verified'; } else { echo 'Installer
corrupt'; unlink('composer-setup.php'); } echo PHP_EOL;"
root@nells-deb8:/var/www# php composer-setup.php
root@nells-deb8:/var/www# php -r "unlink('composer-setup.php');"
```

Damit composer auf dem gesamten System und für alle User verfügbar ist, führen wir folgendes Kommando aus:

```
root@nells-deb8:/var/www# mv composer.phar /usr/local/bin/composer
```



### 3.5.3 Anpassung apache-site

```
root@nells-deb8:/home/nells1# nano /etc/apache2/sites-enabled/000-default.conf
```

Folgende Einträge wurde angepasst:

```
ServerName abc.2488.ch  
ServerAdmin nells1@bfh.ch  
DocumentRoot /var/www/html
```

Abschliessend starten Apache nochmals neu:

```
root@nells-deb8:/home/nells1# service apache2 restart
```

## 4 Daten aus GIT beziehen

- Daten von GIT herunterladen, verschieben
- Abhängigkeiten mit Composer installieren (# composer install)
- Berechtigungen kontrollieren und ggf. mit *chmod* und *chown* korrigieren

## 5 Datenbank vorbereiten

Datenbank erstellen, SQL-Skripte ausführen (z.B. mit phpMyAdmin)

## E. JSON-Beispieldateien

```

export.js.js
Dienstag, 17. Januar 2017 10:48

<!-- parameters.json -->
{
  "g0": "3582668410913453...",
  "g1": "2773228543134866...",
  "h0": "1843417255026200...",
  "h1": "1015144610848823...",
  "h2": "1098555030594512...",
  "o": "4044800088407278...",
  "p": "1248395089014592..."
}

<!-- Payload_register-new-voter.json -->
{
  "email": "bob@bfh.ch",
  "publicCredential": "633111441323...",
  "appVersion": "1.15"
}

<!-- JWS_register-new-voter.json -->
{
  "signature":
  "eyJhbGciOiJSUzI1NiJ9eyJlbWFpbCI6ImJvYkBiZmguy2giLCJwdWJsaWNDcmVkJzW50aWFsIjoiNjMzMTExNDQzMzIzMTU0OTg4NTk0MDU4OTQ1MTIxNDk5NzU0OTUzMzA4OTI4ODY3NDk5MTc1OTE2OTY2Mzc2MDkxNjExODE2MjMzNDMzNzk0NDA3NDI1NTAyODUxNDk0NDQ5NjM3NzK5ODEzNTYyMDk5Njg1ODUyNjIzMzcyNzk1NjYwNjczNzQ2ODM2NjAxNDc0NDg2NDI4NzM2MDAxNzkxNTM0NDc5NjYwOTM4ODA3MzgxMzQxNTE0MTA2OTg1OTg4NDE2MjQ1OTQxODEzODAyMDM3MTE1NjcxMDQyMjEzMzEyODAyOTYxODUwMTg3ODU2MzMxOTMxOTMyNDY4Mjg0NTA2NDExMTY2NDc3MjU4ODE2NDAzNzkxMzU1OTEYmjc0MjkyNDA0NDMwMDkwMzY4NjciLCJhCHBWZXJzaW9uIjoiMS4xNSJ9.b1y4H4VD9bzTBI3IDxSRUh9JSb44dwGW-cBB1RwO_5z60Sw7Gx4K2dX6oYQUjQ3MnjUrMG_GTtyHOnLLWH0T1hCwTOH5exSmuGFF1xmXiPQeHd6ygie_QaIdJiHy1F13gcfZh1600oPoj0SupqCbhsFdAmeIwEWUoNt0Tie2IHhyc9asq8t2JJUuErud0xcB88Eoszc3GRG6DUPrMeicNOR2YC59PRQMhg9x3e87NPjVwg3y0uy4QBqmeXPfZL5gbBd5J83LgZNERJOQO_iikM ZwPbhKaAInpaEkxFsgX9mY5gnjgFBFssQcSPAUOD5CIme176oCbpJhVphNPtG"
}

<!-- parameters.json -->
[{
  "email": "hector@bfh.ch",
  "publicCredential": "949561781628...",
  "appVersion": "1.15"
}, {
  "email": "alice@bfh.ch",
  "publicCredential": "206751108882...",
  "appVersion": "1.15"
}, {
  "email": "bob@bfh.ch",
  "publicCredential": "274106800616...",
  "appVersion": "1.15"
}]

<!-- Payload_new-election.json -->
{
  "author": "bob@bfh.ch",
  "electionTitle": "Demo-Wahl",
  "beginDate": "2017-01-16 08:00:00",
  "endDate": "2017-01-22 17:00:00",
  "appVersion": "0.15",
  "coefficients": "134095074800...",
  "electionGenerator": "351832591783...",
  "votingTopic": {
    "topic": "What do you think about my demo?",
    "pick": 1,
    "options": ["Yes", "No"]
  },
  "voters": [
    {
      "email": "alice@bfh.ch",
      "publicCredential": "704795796009...",
      "appVersion": "1.15"
    },
    {
      "email": "bob@bfh.ch",
      "publicCredential": "274106800616...",
      "appVersion": "1.15"
    }
  ]
}

```

```

}

<!-- JWS_new-election.json -->
{
  "signature": "eyJhbGciOiJSUzI1NiJ9.eyJhdXRob3iOiJib2JAYmZoLmNoIiwiZWx1Y3Rpb25UaXrsZSI6IkRlbW8tV2FobCIsImJlZ2luRGF0ZSI6IjIwMtctMDEtMTYgMDA6MDA6MDAiLCJbmREYXR1IjoimjAxNy0wMS0yMiAwMDowMDowMCIsImFwcFZlcnNpb24iOiIwLjE1IiwiY291ZmZpY211bnRzIjoimTM0MDk1MDc0ODAwNzg0MjEjNtkeyODM0ODA0MDMzMjK0ODM5OTkzNzM4NTU1ODg2MjKzNtk1Nzg2MDQwNjE3NzIwNDE0MTIyMzQxNDUxMTM4ODI1Njg4NTA4NTU3MzQ5ODY2Nzc1OTk1MjY5NzMyMzgxNDcxOTYyMzk4NzI5NzI4NjA1OTE5NjMzNzgxMzI1NTQyMtg00TA5MjUzMzc4MzI3MTgyOTI3NzYxMzM3NDQ4NTM3NDQ1MDEyMTQ4MTQ3Njcg0NzK0NjkzNtQwOTM1MjA0NjgZMzc4NTUxMzE3MDI1OTg0MTM2MTQxMjU3OTIyODI1OTc1MTM1NDU0MzgxOTc4OTk1ODc4ODAxMjgwNzgZNDAXODg0MjY5NzU4OTg0NjI5MTC5MjKxNjI4MjU1OTUxOTc5NDQzMjQ5NDE0ODk3OTI3MzUwNTI5MTEzMTQwMDcyODM5NDU4ODUxMTc0Ndc2NDM1NTI4NTMxODA4OTM00Tc1NzU2NTY2NzI5ODU3NzU4NjgyMDgzMjQ2Ntg4NzM0MDAyOTczMzMyOTU1MDcwMDIyMDM1MjY5MTE4NTewMTk3NTgzNDEyMTA3MzA0Mz0MDA1NDEyMTQ0NTA5NDQ0MTI5OTI5NDE2Njg4OTM0MzI4NTU2MDAyMjQxMzk4MzMyNjK2NzA1NTQwDgxMjI3NzI2NzUxMjI3MTU5DcyMjE4NjAxNzK3NjEyOTM5NDgxMjI3ODgZnje2NDU4ODMxMDM5NDcxMDY4NzAyNzcxMjKxMDE0MTQ4MTA5MjY5Mtg0NTMzOTMzNTE4NTM0MjK0NzU2NjE0MjQxMjMwMTI2ODY4MzU1MDIzMjgwNTA1ODQzNDYzNzgZnja2MjI1MjQ4Mjcg0MjkwNTU2NzC5MjMxMDQ1MjY2ODYYMTczNzQyNTU5NjY3MjU2MzAzMjY1NTE2MzA5MjE00TM4MzM0MjUwOTywNDEzNjY3NzExMzM5Mtg4MDUzMDYwNTY0MDcxNzK2OTgymjYyMTU2NzUzNzM4OTQ4NTI3MzA0NTQ2NjC3NDA1OTg3NDI4MzMzI1NDg0MDEyNDExNjgyMtg50TQyOTkwNTk1ODIxMDA4NzI3MzA1NjQ3NjAxODYyMzE5NzAxNzY1OTcyODY5Mtg4NDAyMDgymjY4MDA0MTY5MzMyNDgxNTExNjYwOTI4ODQ1MDAxODMwMTU00Tc2NjewNDM4OTUzMyMjE0MjNwNjC1OTI5NjM1MTk2NzY4NzE5OTg4MzcmwMDgxNTy2MzU2NTQxMDgymjg3MDQyNDg4MDkwOTcxOTc4NTUyNzK1NTYzMTU0NzExNzAyOTyMjI1MDAxMTA4MTQ1ODg5MDM3OTU3NTk1MzMONjczNz5NjYzMDA0NzC2OTA1OTY3NDk3NzQ4MDA0NzQ3OTQ5NjAyMjAzMTI2NTc0MTkwNDY4NjI4NjExODgxNTcyNDM1NTY3NjUxMDYzMTUyNTU0NTExNjI5OTQ4NjI0MzA3Mjcg0Tc00TeMDY3NjAyOTQ5MzA1NDY2MDE5ODcxOTA4OTazNjY5MTC1MDMwMTgymzgymzgxNzYxNzC1NjC2MTA1MzYyMDA5NtIyODIzOTI5OTU5NzE4Mtg50Dc4Njcg4NDQzMzYwOTc0MDg3MTE5MTg4NTQwNTU3NjgwNjAxMDIxMDYyNjE5MjI4NDY3ODE5MjU0NDk1Nta3NjE1NjUzNjMzNjUwNzEwNzQ2NjI00Tm0NTgymjMwNTU5MzkwOTE2Mzk5MzgxmjkyNzkwNjgMzTmkyNdk1NjUwMjCwMjC1NzA5Mz5MNDI0NDEzNxDmxtM3MzUxNTQ3MDI1NTAyODM1OTUxMTU4ODkwNmwMTE1MDA5MD51Mz4NzgwyNT0Tg3OTCzNzMyMzcxMDgxOTc2DoxA0Tg4MjExNdc0NjY1MTgNTA2ODYxNTYyNDI3OTU5NjgxNzkwOTQ50TmWzI2OTg4MzQwNdg5Mta50T10NDI1NjE2MDA5MjE3ODM0MjM5MjE2NjA1ODYxNDU0NjK1MzU00Dk2MDY5NTU3ODY2MzU3MjEwNTI5MjMzNzY1NDYyNtC0NDEzMjY3ODEyMzC5MDkzODQ5MTY2MzY2NDUxMDkyNDE3NDU3OTyNzg2MDUyMTU5MjK1NzU4ODE2NzYwMz3NzC1MzgyNzA1MzA5NDI1MtaZOTcwMzg4NDgxODQ2OTE5MjQ1NjEwMzC3NjU2NzY3MTY3Mjg5NTgxMzc4Nzg0MDk3MDMwNzK1NjAzMjEwNjY1IMDU5MzU2NDUy0DQ2Mjg5MDA3MTU0NTQ0MDg4MTg1ODAwMzUwODU1MTM1NzkyNjgxOTE4MjYzMzg1Nzg5MjMwNjM1ODU2MTQ1MzI5ODg2MDQ2ODYxMzK5NzE2MjQ0NDkxNDgxOTYyNDU1MjAwOTgxmjgyMjgNzAwNjI0NTkONzUwMDIwMTQ0MDczMjC1OTMzMDkxNTcwNTgZMtyyMDcxOTA0MTE1NTE0ODY3NzU5MzAwNzQwNtI0NzYxNdkzMDazMdc5Ndg0MDU0MzK2NTUxODcxNjE0OTy0MzczNTEzMzQ4NjE4OTU3MDM3NTI5MDA2NjQyOTkyMDMzMy2MjKxMDk5NzUzMz5NzK5NjYxNDYzNTQyNTI1NTAwMTg4MjCwODIxMzU3NzC0MDU3MDIxNjKwNTA2OTIwMTM0NDQ2MDA4NjA5Nz0MzA0TgNjC5Mtk2NzK5Ntm4Ntc1NtuwMzU3NjE5ODQzNTM00DzIzNzYwNjk5Ntm4Ndg0MDM3Mtg5MTU2NTc2NDYzODU5MjAyNjQ3MTM5NDI4MzQ1Mjg0MjI1MTY2MTQzMDA4NTQ00TA1OTQ3OTIwNTA2ODY0MjA4Njcg3OTMymjQzNTEyNzY3OTyNjK5NDE2NDg5ODc0MDc3Nzg2NTIwMjYzMjC2MzIwNDI2MDU3NTy0MzI4MTIyODM2MjUwNjU0NjK5MTY0NTk1NDM3NU1MDExNDE4NTI1NzI4OTU3NDg3NTY0NzEyOTE0NzMxMDY4OTIzOTI2NjczNDkyNTE1NTIzNjEwNjK1ODA2MjQyMjY1NjCzMTA5NDY0MTI0MjcyNDA0MDk0MjY4MjI5NjQxOTcyOTQ4MjK3NjI4NzQyMTA4NDI4OTAzNDg0NDkxMDg1NDA5Ntc3NDAzMTE4OTkwNDE3MTQyMzU0NTE1NDIwNDA4MDU1OTkzMzQxNTQ50Dk4MTM00Dk20DczMtkyMzg0OTk0ODM1MTM2NjKwMjC2MjA0NzK1NTE0NTc0NTU5IiwiZwX1Y3Rpb25Hzw51cmF0b3Ii0iIzNTE4MzI1Ote3ODM3MjM2NzQ0NzE1NjY2MzI3NDI4NTA0NTU5NDcyNzczNTM3NTg5Mtc3Mzc20MzNzI4MTA0NjI0NTyWMDU1NTM4Mdg4Mzg4NjU4NT1MjE4OTUyOTg2MTUxNjMxOTy30DkxMtzQ1MzQwMjEwODU1NDg4NTg5NjMwNdczMTcxNDA20Dc4MTM0NDcyMTE4NjQ4OTQzMDA0MjE5NjI1ODMzNzE3NzEwMjU2Nzg5NjK1MDYzMTU5NDg4MjExNTy2OTMzODYwNzgZntu4MzAzMzA3MjK2MjQxNjgyMzg1Nta1NTMzOTQ1NjQzNtIxmDewNDY1NTE1MDY3MDg1MTkzMTc1MzkwOTuNzKzNDM2MDI3MTIwMTI5MjgwMDM1NjA2NTI2MDYyNzg3MTY0NCIsInZvdGluZ1RvcG1jIjg7InRvcG1jIjoiV2hhdCkbbyB5b3UgdGhpbmgsyWJvdXQgbXkgZGvtbz8iLCJwaWnrIjoxLCJvcHrpB25zIjpb1lllcIyIsIk5v119LcJ2b3R1cmMi0lt7ImVtYwlsIjoiYwpxpY2VAYmZoLmNoIiwiCHVibG1jQ3J1ZGVudG1hbC16IjcwNdc5Ntc5NjAwotkyMjE3MzE50tg5MDY3MzA1MzIxNTQyNDQ3NjIwNzU1MtpyMzgymjQyNzgZODUzODk2NDI00TQwNjE2NzM1MTIxNTIxMTE10Dy1Ndc50DcyNjC5Mj20DE3NDcyMTI1MjA1NjEwMDkxNjQwNDEyOTazNjK4NDQ3NjMyotk20T2MD02D0U10Dy4NTQ4MzK3MDEzNDQ4MTg2MjkyODMyOTyNDExODg0NTk4Nt95NjY3NDQ3NzMxMjE4NTcwOTuNjI0NzQ0MzMyMtk4MjC4MTI5NjU5ODU1MDU1MzA2NTI4NzM1NzQyMzg0MTgwNTIxODYwMDc5MTCyNzC1NzA00TE3MDE4NDcwNTUzMDQyOTc40TgZMTA5OTy1NTMzMTQ1MzkwOTI0NjQ3IiwiYXBwVmVyc21vbiI6IjEuMTUifSx7ImVtYwlsIjoiYm9iQGJmaC5jacIsInB1YmxpY0NyZWRlbnRpYyWwi0iIyNzQxMDY4MDA2MTY0NzA3NjI5MjC5NjQzOTYzMTQyNTg3NzQxMjkyMTcwtM1yMzkxMDEyNjY0ODQ1OTyNjcyMjU4MzK2MjY2OTuZNDg3MzUyNzgxmzQyNjK5NTYyMjQ4NTczOTEwMDUwNjC50Tg4MTU4Njg2MjExMTY3NDkxNTk4NTQyNDkwNTc1Nzg0NTA3MDk0Mzk0Njg5MjcyMzC4NjY4ODA10DMyMTkwODY2NTE2MDI5NjUzODk0NTcyNDg1MDM0MzYyODI4NzKzMzQ20TczOTQxOTuNjYxNTc4MjK5MTk5MzIyNDExNTA4NjY4MDMwMzC0Njcg5MDAwMzY3MTAxNzA2NjUyNzY1ODE5NjUwNTA2NjA4ODM0NTc4MDk50TmWNDQ5MTk5MTA5NzI0MTE3NSIsImFwcFZlcnNpb24i0iIxLjE1In1dfQ.bK7-augT1MaubGqsRpjz4X51nZle3rs0sRi00fEVA13K-03NyHj8X3xKrbBGrJyDRsbuzudHSyBrx7mI47qszd6rb4c2k5HjfxVr6MoUI-QwcbFN915S4NkkPw83nuPg8kbIH1YktXT9PBW4bsMOONwesPT6NAvgudMyq3NhLmBeN1jpaAaPFUMyN-b9gCb-HBL2zfp_8WM2vcBGLnfWFZxqesLGRMdsp1CgYu-vI9NK5NgoFt4bK94HjfrEoaYccSEEMwhLkTL80RzNqQug7EEaLbfDfcOYybFec7a87us4VTOOOIHrevOj4_7s4gyvrmNocrz5Ezz6ztK9N0tVb5w"
}

```

```

export.js.js
Dienstag, 17. Januar 2017 10:48

<!-- open-elections-headers -->
[{
  "id": "54",
  "electionTitle": "Demo-Wahl",
  "beginDate": "2017-01-16 08:00:00",
  "endDate": "2017-01-22 17:00:00"
}]

<!-- election_by_id.json -->
{
  "author": "bob@bfh.ch",
  "electionTitle": "Demo-Wahl",
  "beginDate": "2017-01-16 08:00:00",
  "endDate": "2017-01-22 17:00:00",
  "appVersion": "0.15",
  "coefficients": "134095074800...",
  "electionGenerator": "351832591783",
  "votingTopic": {
    "topic": "What do you think about my demo?",
    "pick": 1,
    "options": ["Yes", "No"]
  },
  "voters": [
    {
      "email": "alice@bfh.ch",
      "publicCredential": "704795796009...",
      "appVersion": "1.15"
    },
    {
      "email": "bob@bfh.ch",
      "publicCredential": "274106800616...",
      "appVersion": "1.15"
    }
  ]
}

<!-- ballot.json -->
{
  "e": ["Yes"],
  "u_Hat": "569031786385...",
  "c": "148712772381...",
  "d": "209525745158...",
  "pi1": "[[\"139176951188...\"|[\"121257891365...\"|\"371401529778...\"]]]",
  "pi2": "[[\"268871424069...\"|[\"436849469132...\"|\"242162499728...\"|\"325535572881...\"|\"342460773663...\"|[\"200636351030...\"|\"355018256115...\"]]]",
  "pi3": "[[\"430809301995...\"|\"549478957410...\"|\"112711760857...\"]]"
}

<!-- Payload_result.json -->
{
  "author": "bob@bfh.ch",
  "result": [
    {
      "topic": "Want to use Tor?",
      "pick": 1,
      "options": [
        {
          "optTitle": "Yes",
          "count": 0
        },
        {
          "optTitle": "No",
          "count": 1
        }
      ]
    },
    "ballots": [
      {
        "id": 89,
        "e": ["No"],
        "valid": true,
        "reason": "-"
      },
      {
        "id": 90,
        "e": ["No"],
        "valid": false,
        "reason": "Already selected another vote of the same voter"
      }
    ]
}

```

```

        "id": 91,
        "e": ["Yes"],
        "valid": false,
        "reason": "Already selected another vote of the same voter"
    }, {
        "id": 92,
        "e": ["Yes"],
        "valid": false,
        "reason": "Already selected another vote of the same voter"
    }, {
        "id": 93,
        "e": ["Yes"],
        "valid": false,
        "reason": "Already selected another vote of the same voter"
    }, {
        "id": 94,
        "e": ["Yes"],
        "valid": false,
        "reason": "Already selected another vote of the same voter"
    }, {
        "id": 95,
        "e": ["No"],
        "valid": false,
        "reason": "Already selected another vote of the same voter"
    }, {
        "id": 96,
        "e": ["Yes"],
        "valid": false,
        "reason": "Already selected another vote of the same voter"
    }
}

<!-- JWS_result.json -->
{
    "signature":
"eyJhbGciOiJSUzI1NiJ9.eyJhdXR0b3IiOjIpb2JAYmZoLmNoIiwicmVzdWx0IjpbeeyJ0b3BpYyI6IldhbnQgdG8g
dXNlIFRvcj8iLCJwaWNrIjoxLCJvcHRpb25zIjpbeeyJvcHRUaXRsZSI6I111cyIsImNvdW50IjowfSx7Im9wdFRpdG
xIiJoiTm8iLCJjb3VudCI6MX1dfV0sImJhbGxvdHMiOlt7ImlkIjo4OSwiZSI6WyJ0byJdLCJ2YWxpZCI6dHJ1ZSwi
cmVhc29uIjoiLSJ9LHsiaWQiOjkwLCJ1IpbIk5vI10sInZhbG1kIjpmYWxzzSwicmVhc29uIjoiQWxyZWFKeSBzzW
xIy3R1ZCBhb90aGVyIHZvdGUgb2YgdGh1IHNhbWUgdm90ZXIfSx7ImlkIjo5MSwiZSI6WyJZZXMiXSwidmFsaWQi
OmZhbHN1LCJyZWFzb24iOjBbHJ1YWR5IHN1bGVjdGVkIGFub3RoZXIgdm90ZSBvZiB0aGUgc2FtZSB2b3R1ciJ9LH
siaWQiOjkyLCJ1IjpbI111cyJdLCJ2YWxpZCI6ZmFsc2UsInJ1YXNvbI61kFscmVhZHkgc2VsZWN0ZWQgYW5vdGh1
ciB2b3R1IG9mIHRoZSBzYW11IHZvdGVyIn0seyJpZCI6OTMsImUiOlsvWWVzI10sInZhbG1kIjpmYWxzzSwicmVhc2
9uIjoiQWxyZWFKeSBzZw1Y3R1ZCBhb90aGVyIHZvdGUgb2YgdGh1IHNhbWUgdm90ZXIfSx7ImlkIjo5NCwiZSI6
WyJZZXMiXSwidmFsaWQiOmZhbHN1LCJyZWFzb24iOjBbHJ1YWR5IHN1bGVjdGVkIGFub3RoZXIgdm90ZSBvZiB0aG
Ugc2FtZSB2b3R1ciJ9LHsiaWQiOjk1LCJ1IjpbIk5vI10sInZhbG1kIjpmYWxzzSwicmVhc29uIjoiQWxyZWFKeSBz
Zw1Y3R1ZCBhb90aGVyIHZvdGUgb2YgdGh1IHNhbWUgdm90ZXIfSx7ImlkIjo5NiwiZSI6WyJZZXMiXSwidmFsaW
QiOmZhbHN1LCJyZWFzb24iOjBbHJ1YWR5IHN1bGVjdGVkIGFub3RoZXIgdm90ZSBvZiB0aGUgc2FtZSB2b3R1ciJ9
XX0.dtRMGjNU0j4sTo3pfVBelAg2jGp3I70dTzhZ3iXF_uh_fkk103d7S99JKEbfuMjFgOCj1FIuhrNGg414sKiak6
4QYHzbu0c58q09ss-7AdfUHU_a3jWS7MFH30zZWmon7U_9YBsbFJ2-jRKdnaEYhIDU6Fi3EPFWU2yjjJqgzxUra4PP
Sm_ZOB-wy21ZT_amuPKWQeLiLCuWFWwYZG344xmyuj35JMjF_IYjQhS78rzV_g9TAcB-vmCag3fBqpJHoaw5Pjr78
5qins-GIsX1TpC2WAzBOtn7noqk60yoNSUufbnOU4CQrbyMKlnmQC1qwX1KLExnqIEBk4a46z-rg"
}

```



## F. Test Cases



# 1 Systemtest

## Bemerkung:

Damit der Test nicht auf das Vorhandensein bestehender Daten angewiesen ist, müssen die Testfälle der Reihe nach bearbeitet werden. Dabei dient das Ergebnis aus dem vorangegangenen Fall jeweils als Daten für den nächsten Fall.

Nr. und Name:	1: Wähler Registrierung
Szenario:	Der Wähler registriert sich auf der elektronischen Abstimmungsplattform.
Kurzbeschreibung:	Die Wähleridentität des Wählers wird an das Bulletin-Board gesendet.
Voraussetzung:	Für diesen Schritt muss auf der Test-Maschine ein Signierungszertifikat und der dazu passende private Key in einem Java Keystore lokal abgelegt sein. Das Zertifikat muss zudem in der PKI des Bulletin Boards hinterlegt werden. Zusätzlich muss die E-Mail-Adresse des Zertifikats dem Tester zu Beginn des Tests bekannt gegeben werden.
Ergebnisse / Nachbedingung:	Das Bulletin-Board kennt das authentisierte/signierte Wähler-Credential des Wählers.

## Ablauf:

Nr.	Was
1.01	Rufen Sie vor dem Öffnen der VoterApp die Webseite <a href="https://abc.2488.ch/view/voters">https://abc.2488.ch/view/voters</a> auf. Diese zeigt eine Liste aller zurzeit registrierten Wählern
1.02	Öffnen Sie die VoterApp über die ausführbare Datei oder eine Verknüpfung.
1.03	Wählen Sie «Register», um den Registrierungsprozess zu starten.
1.04	Sie werden nun aufgefordert eine E-Mail-Adresse anzugeben. Verwenden Sie hierfür die zu Beginn des Tests erhaltene E-Mail-Adresse.
1.05	Bestätigen Sie Ihre Eingabe mit «Create». Sie werden nun zum Startbildschirm der VoterApp weitergeleitet.
1.06	Rufen Sie erneut die Webseite <a href="https://abc.2488.ch/view/voters">https://abc.2488.ch/view/voters</a> auf und stellen Sie sicher, dass ein neuer Eintrag mit der von Ihnen eingegebenen Email-Adresse hinzugekommen ist.



Nr. und Name:	2: Wahl Definition
Szenario:	Die Wahladministration definiert eine neue Wahl.
Kurzbeschreibung:	Die Wahladministration erstellt eine neue Wahl und definiert die Rahmenbedingungen sowie die zugelassenen Wähler.
Voraussetzung:	Für diesen Schritt muss auf der Test-Maschine ein Signierungszertifikat und der dazu passende private Key in einem Java Keystore lokal abgelegt sein. Das Zertifikat muss zudem in der PKI des Bulletin Boards hinterlegt werden. Zusätzlich muss der registrierte Benutzer aus dem Vortest benötigt.
Ergebnisse / Nachbedingung:	Die Wahl ist publiziert und kann durchgeführt werden.

**Ablauf:**

Nr.	Was
2.01	Rufen Sie zum Beginn des Tests die Webseite <a href="https://abc.2488.ch/view/elections">https://abc.2488.ch/view/elections</a> auf. Diese zeigt eine Liste aller erfassten Wahlen.
2.02	Öffnen Sie nun die AdminApp über die ausführbare Datei oder eine Verknüpfung.
2.03	Wählen Sie «Create election», um den Prozess zum Erstellen einer Wahl zu beginnen.
2.04	Sie werden nun aufgefordert einen Titel für die neue Wahl einzugeben. Geben sie hier ein Titel ein, den Sie später wiedererkennen können. Zum Beispiel eine Kombination aus Test, Ihrem Namen und dem aktuellen Datum. Notieren Sie sich den eingegebenen Titel und bestätigen Sie Ihre Eingabe mit «Next».
2.05	Sie sehen nun die Liste aller registrierten Wähler. Darunter sollte sich auch der Wähler befinden, welchen Sie im ersten Szenario registriert haben. Wählen Sie nun beliebig viele Wähler aus. Stellen Sie aber sicher, dass Sie den von Ihnen erstellten Wähler ausgewählt haben. Bestätigen Sie Ihre Auswahl mit «Next».
2.06	Sie werden nun aufgefordert eine Abstimmungsfrage und zwei Optionen zu erfassen. Geben Sie eine beliebige Abstimmungsfrage und zwei unterschiedliche Optionen ein. Bestätigen Sie Ihre Angaben mit «Next»
2.07	Nun werden Sie aufgefordert die Abstimmungsperiode der Wahl anzugeben. Stellen Sie sicher, dass sich der aktuelle Tag innerhalb der Abstimmungsperiode befindet. Stellen Sie dazu das Ende der Abstimmungsperiode mindestens auf das Datum des nächsten Tages. Bestätigen Sie Ihre Auswahl mit «Next».
2.08	Sie sehen nun eine Zusammenfassung Ihrer Eingaben. Kontrollieren Sie die Angaben und schliessen Sie den Vorgang mit «Create» ab. Sie werden anschliessend auf den Startbildschirm weitergeleitet.
2.09	Rufen Sie die Webseite <a href="https://abc.2488.ch/view/elections">https://abc.2488.ch/view/elections</a> erneut auf und stellen Sie sicher, dass Ihre erstellte Wahl sich nun in der Auflistung befindet. Ihre Wahl können Sie anhand des Titels erkennen, welchen Sie im Schritt 2.04 eingegeben und notiert haben.



Nr. und Name:	3: Stimmabgabe
Szenario:	Der Wähler gibt seine Stimme ab.
Kurzbeschreibung:	Die Stimme („Ballot“) des Wählers wird generiert und an das Bulletin-Board gesendet.
Voraussetzung:	Benötigt werden für dieses Szenario die Produkte aus den beiden vorangegangenen Szenarien. Erstens müssen im lokalen KeyStore registrierte Private Credentials vorhanden sein und zum anderen muss eine laufende Wahl vorhanden sein zu der dieser Wähler zugelassen ist.
Ergebnisse / Nachbedingung:	Die Stimme des Wählers ist auf dem Bulletin-Board eingetragen.

**Ablauf:**

Nr.	Was
3.01	Öffnen Sie die VoterApp über die ausführbare Datei oder eine Verknüpfung.
3.02	Wählen Sie «Vote», um den Abstimmungsprozess zu starten.
3.03	Sie sehen nun eine Liste mit allen zurzeit laufenden Wahlen. In der Liste sollte sich die von Ihnen erfasste Wahl befinden. Diese können Sie durch den Titel erkennen, welchen Sie im beim Erfassen der Wahl eingetragen und notiert haben. Wählen Sie die von Ihnen erstellte Wahl aus und bestätigen Sie Ihre Auswahl mit «Next».
3.04	Nun wird die Abstimmungsfrage angezeigt und Sie werden aufgefordert eine Option auszuwählen, für welche Sie stimmen möchten. Wählen Sie eine der Optionen aus und notieren Sie sie. Bestätigen Sie anschliessend Ihre Wahl mit «Next».
3.05	Sie sehen nun eine Zusammenfassung Ihrer Eingaben. Kontrollieren Sie die Angaben und schliessen Sie den Vorgang mit «Cast Ballot» ab. Anschliessend werden Sie zum Startbildschirm weitergeleitet. Dieser Vorgang kann einige Sekunden dauern.



Nr. und Name:	4: Auszählung – Verifikation
Szenario:	Der Verifikator / Auszählung verifiziert alle Stimmen und berechnet das Ergebnis.
Kurzbeschreibung:	Die Wahl wird nach Abschluss durch den Verifikator / Auszählung überprüft und ausgewertet.
Voraussetzung:	Für diesen Schritt muss auf der Test-Maschine ein Signierungszertifikat und der dazu passende private Key in einem Java Keystore lokal abgelegt sein. Das Zertifikat muss zudem in der PKI des Bulletin Boards hinterlegt werden. Zusätzlich müssen die Produkte aus den beiden vorangegangenen Tests vorhanden sein. Eine Wahl und ein dafür abgegebener Stimmzettel. Der Test muss entweder nach Ablauf der Wahlperiode durchgeführt werden oder aber die Applikation wird für den Test so angepasst, dass auch laufende Wahlen ausgewertet werden können.
Ergebnisse / Nachbedingung:	Das Wahlresultat wurde ausgegeben.

**Ablauf:**

Nr.	Was
4.01	Öffnen Sie die VerifierApp über die ausführbare Datei oder eine Verknüpfung.
4.02	Wählen Sie «Verify election», um den Auszählungsprozess zu starten.
4.03	Sie sehen nun eine Liste mit Wahlen. In der Liste sollte sich die von Ihnen erfasste Wahl befinden. Diese können Sie durch den Titel erkennen, welchen Sie im beim Erfassen der Wahl eingetragen und notiert haben. Wählen Sie die von Ihnen erstellte Wahl aus und bestätigen Sie Ihre Auswahl mit «Result». Das Resultat der Wahl wird nun berechnet. Dies kann einige Sekunden dauern
4.04	Sie sehen nun das Ergebnis Ihrer Wahl. Stellen Sie sicher, dass das Ergebnis der Wahl mit der von Ihnen abgegebenen Stimme übereinstimmt. Bestätigen Sie das Resultat mit «Publish». Sie werden nun auf den Titelbildschirm weitergeleitet.



## **G. Sitzungsprotokolle**



**Berner Fachhochschule**  
BFH-TI

Sebastian Nellen  
Student  
Merkurstrasse 20  
3613 Steffisburg  
Telefon +41 79 737 57 52  
sebastian.nellen@students.bfh.ch

## Protokoll

Termin: Dienstag, 20. September 2016, 11:00 Uhr  
Ort: BFH, Rolex-Gebäude, Biel  
Anwesend: Haenni Rolf (Betreuer),  
Locher Philipp (Betreuer),  
Bürk Timo (Student),  
Nellen Sebastian (Student)

### Kick-Off Bachelor-Thesis

#### Traktanden

1. Deliverables und Termine
2. Treffen von Betreuer und Studierenden
3. Weiteres Vorgehen
4. Diverses

#### 1. Deliverables und Termine

- Konkrete Aufgabenstellung
  - o Generell: E-Voting Schema aus Projekt 2 in einer Form X implementieren
  - o Implementation in Java, GUI mit JavaFX (entweder via WebView oder JavaFX)
  - o **Teil 1: Wahl durchführen**
    - *Komponente 1:*  
Admin-App zur Wahl-Definition
      - Wahlverfahren: z.B. 1 aus 2, 1 aus n, k aus n
      - n muss festgelegt werden können
      - Wahl organisieren (Kandidaten definieren)
      - Wählerliste eingeben
    - *Komponente 2:*  
Client für Stimmabgabe und Registrierung (Java, Unicrypt)
    - *Komponente 3:*  
Bulletin-Board
    - *Komponente 4:*  
Wahl auszählen und verifizieren (Java, Unicrypt)
  - o **Teil 2: Anonymer Kanal**
    - MUST: Recherche betr. Machbarkeit
      - Z.B. Stimmabgabe über TOR-Netzwerk
    - CAN: je nach je Implementation
      - Ohne Service, unsere App müsste dies integriert haben
      - Möglichkeiten, Libraries aufzeigen
  - o **Teil 3: Append-only bulletin-board**
    - *MUST-Ansatz:*  
MySQL-DB mit REST-Schnittstelle
    - *CAN-Ansatz:*  
Vorhandene Implementation der BFH nutzen (Severin Hauser)

- **Teil 4: Votes**
  - Verschlüsselung der Votes (mit Trusted Authority)
  - Private-Key (Shared Keys, Key-Management) → Trustee-App
  - Varianten
    - Nicht verschlüsselt
    - 1 Key pro Wahl mit vertrauenswürdiger Instanz
    - Shared Keys oder Entschlüsselung - so kann derselbe Private Key für mehrere Wahlen verwenden
    - mit Threshold (verteilte Schlüssel)
- Technisch
  - Key-Management ist ein MUST, da praxis-relevant
    - Pragmatischer Ansatz:  
Public-Credential mit PublicKey von XY signieren  
**Wir stützen uns auf vorhandene PKI ab.**  
Admin muss PKI X oder Y vertrauen
    - Zertifikate-Handling via Java
  - Zu klären:
    - Wahlperiode (Konsequenz von zu früh oder zu spät?!)
      - Fragen zu Timestamps, Veränderungen
    - Dezentrales System, nur Bulletin Board online
    - Was passiert mit falschen Stimmen oder Duplikaten?
    - Wahl-Identifier (unique), für verschiedene Wahlen
- Deliverables
  - Schriftliche Dokumentation
    - Abgabe-Termin: 19. Januar 2017, 23:59 Uhr (Vorabend Finaltag)
  - Draft der Arbeit ca. 2 Wochen vor Abgabe
  - Zeitplan
  - Tests und Use Cases

## 2. Treffen von Betreuer und Studierenden

- Aus Sicht von uns Studenten hat sich der 2-Wochen-Rhythmus bewährt
- Regelmässige Treffen: in Biel, am Dienstag, zwischen 11:00 Uhr und 12:00 Uhr
- Nächste Treffen:
  - Dienstag, 27. September 2016 um 11:00 Uhr

## 3. Weiteres Vorgehen

- Projekt-Planung
- Applikations-Design
  - Funktionsumfang
  - Tests

## 4. Diverses

- Unterschreiben Bachelorthesis-Aufgabe durch Betreuer
- Experten-Meeting mit Planungs-Dokumenten
  - z.B. Use Cases (Wahl durchführen, Prototyp machen)
  - Milestone-Planung/Zeitplan
- Tests (Unit-Tests, GUI-Tests mit x Test-Cases) als Teil der Planung
- LIZENZ unseres Codes = Dual (AGPL und Commercial, analog Unicrypt), Ablage (öffentliches GIT, private)



Für das Protokoll

Verfasser: Sebastian Nellen. Biel, 20.09.16



Berner Fachhochschule  
BFH-TI

Sebastian Nellen  
Student  
Merkurstrasse 20  
3613 Steffisburg  
Telefon +41 79 737 57 52  
sebastian.nellen@students.bfh.ch

## Protokoll

Termin: Dienstag, 27. September 2016, 11:00 Uhr  
Ort: BFH, Rolex-Gebäude, Biel  
Anwesend: Haenni Rolf (Betreuer),  
Locher Philipp (Betreuer),  
Bürk Timo (Student),  
Nellen Sebastian (Student)

### Abstimmung Bachelor-Thesis

#### Traktanden

1. Besprechung Aufgabenstellung (Muss-/Kann-Anforderungen)
  2. Besprechung Zeitplan
  3. Weiteres Vorgehen
  4. Diverses
- 
1. **Besprechung Aufgabenstellung (Muss-/Kann-Anforderungen)**
    - Besprechung «..../50\_Arbeitsverzeichnis/20160926/Planung.xlsx» > „Muss-Kann“
    - Ergänzungen direkt In-File
  2. **Besprechung Zeitplan**
    - Besprechung «..../50\_Arbeitsverzeichnis/20160926/Planung.xlsx» > „Planung“
    - Ergänzungen direkt In-File
  3. **Weiteres Vorgehen**
    - Nächste Treffen:
      - o Dienstag, 11. Oktober 2016 um 8:30 Uhr
  4. **Diverses**
    - LIZENZ unseres Codes = Dual (AGPL und Commercial, analog Unicrypt)
    - Code-Ablage: Github, evg-Student-Repo
    - Daten-Ablage: Dropbox-Share, Einladung an Betreuer versenden (SNe)
    - Tipps:
      - o Wir sollten uns schon vorgängig mal Gedanken zu anonymen Kanal machen, nicht erst in Woche 12.
      - o Wir sollten schon mit Dokumentation anfangen und jeweils Stichwörter festhalten.
      - o Milestones planen
        - z.B. funktionales GUI zuerst für Admin und Client

Für das Protokoll

Verfasser: Sebastian Nellen. Biel, 27.09.16



Berner Fachhochschule  
BFH-TI

Sebastian Nellen  
Student  
Merkurstrasse 20  
3613 Steffisburg  
Telefon +41 79 737 57 52  
sebastian.nellen@students.bfh.ch

## Protokoll

Termin: Dienstag, 11. Oktober 2016, 08:30 Uhr  
Ort: BFH, Rolex-Gebäude, Biel  
Anwesend: Haenni Rolf (Betreuer),  
Locher Philipp (Betreuer),  
Bürk Timo (Student),  
Nellen Sebastian (Student)

### Abstimmung Bachelor-Thesis

#### Traktanden

1. Besprechung Use Cases
2. Besprechung Milestones
3. Besprechung Technologien
4. Diverses
5. Weiteres Vorgehen

#### 1. Besprechung Use Cases

- 4 Use Cases wurden formuliert
  - o 1: Wähler Registrierung
  - o 2: Wahl Definition
  - o 3: Stimmabgabe
  - o 4: Auszählung - Verifikation
- Das Feedback von Herr Locher wird in die überarbeitete Version 2 (.v02) einfließen.

#### 2. Besprechung Milestones

<u>Milestones</u>	
<u>Woche</u>	
1	Kickoff
5	Ready to Implement
7	Wahl kann erstellt werden (Admin-Tool, Bulletin-Board)
9	Stimmabgabe kann durchgeführt werden (Client-App, Bulletin-Board)
11	Wahlresultat kann berechnet und verifiziert werden (Verifikator-Tool, Bulletin-Board)
14 (bzw. Wiederholung)	Abgabe Draft der Dokumentation
16	Abgabe Dokumentation

- Feedback
  - o Kein Web-GUI für Bulletin-Board nötig. Es sollte sehr rudimentär gehalten werden.
  - o Das Bulletin-Board erscheint hier durch mehrfache Nennung als sehr wichtig.
- Bemerkung: Die Milestones sind jeweils Ende Woche X resp. bis Ende Woche X erreicht.

### 3. Besprechung Technologien

Technologien · Kommunikation über JSON

- IDE : Netbeans
- Java 8 mit JavaFX für GUI
- Unicrypt-Library
- Github für SCM
- MySQL-DB für Bulletin-Board

- Feedback
  - o Neben der Datenbank, welche für das Bulletin-Board verwendet wird, ist es noch notwendig, dass erwähnt wird, welche Technologie für die REST-Schnittstelle verwendet wird. Da die Studenten bereits das PHP-Framework Slim kennen, bietet sich dieses an.

### 4. Diverses

- Offene Fragen:
  - o Erster Kontakt zwischen Betreuer und Experte bereits erfolgt? Allfällige Informationen über Experten?
    - → Studenten nehmen Kontakt mit Experte auf
- Sind Probleme mit Unicrypt und Java 8 1.8.0\_101 resp. 1.8.0\_102 bekannt?
  - o Systeme: Mac OS X 10.11 El Capitan & Windows 10
  - o Netbeans 8.1 / 8.2
  - o Klasse *Tupel.java* in *ch.bfh.unicrypt.math.algebra.general.classes*
  - o Meldung:

```
name clash: getInstance(DenseArray<Element>) and
getInstance(DenseArray<? extends Element>) have the same erasure
```

### 5. Weiteres Vorgehen

- Die Studenten setzen sich mit Herrn Flueckiger (Experte) in Verbindung
- Die Studenten bestellen einen Virtual Host bei IT-Services.
- Herr Haenni schaut sich den oben erwähnten Issue betr. der Klasse *Tupel.java* an
- Nächstes Treffen:
  - o Dienstag, 25. Oktober 2016 um 11:00 Uhr



Für das Protokoll

Verfasser: Sebastian Nellen. Biel, 11.10.16



Berner Fachhochschule  
BFH-TI

Sebastian Nellen  
Student  
Merkurstrasse 20  
3613 Steffisburg  
Telefon +41 79 737 57 52  
sebastian.nellen@students.bfh.ch

## Protokoll

Termin: Dienstag, 25. Oktober 2016, 11:00 Uhr  
Ort: BFH, Rolex-Gebäude, Biel  
Anwesend: Haenni Rolf (Betreuer),  
Locher Philipp (Betreuer),  
Bürk Timo (Student),  
Nellen Sebastian (Student)

### Abstimmung Bachelor-Thesis

#### Traktanden

1. Kontakt mit Experte
2. Besprechung Schnittstellen-Diagramm
3. Besprechung GIT-Struktur
4. Diverses
5. Weiteres Vorgehen

#### 1. Kontakt mit Experte

Wir haben uns wie besprochen am 11.10. an Herrn Flueckiger, unseren Experten, gewendet und seither kein Feedback erhalten.

Wir haben nicht nachgefragt, da gemäss letzter Besprechung, Herr Haenni nachfragen wollte Ende jener Woche und wir Doppelspurigkeiten vermeiden wollten.

→ Studenten fragen nach.

→ Treffen mit Experte am Dienstag, 01. November 2016 in Bern

#### 2. Besprechung Schnittstellen-Diagramm

Im Dokument «Schnittstellen-Schema.pdf» im Pfad „7321\_Bachelor-Thesis > 50\_Arbeitsverzeichnis > 20161018“ haben wir die verschiedenen REST-Schnittstellen und Kommunikations-Abläufe festgehalten.

#### 3. Besprechung GIT-Struktur

- Wir haben uns für folgende Struktur entschieden:

**abcVote** (root-Directory)

**adminApp** (Maven-Java-Projekt)

**bulletinBoard** (PHP-Projekt)

**voterApp** (Maven-Java-Projekt)

**verifierApp** (Maven-Java-Projekt)

**utils** (Maven-Java-Projekt, Gemeinsam genutzte Module, Bibliotheken, etc.)

→ Komponentennamen sollten durchgängig sein! Anpassung in Use Cases erfolgt.

→ Die einzelnen Maven-Projekte enthalten neben Source Code auch Tests

→ Es darf keine Abhängigkeit von **utils** zu den anderen Komponenten bestehen

→ **bulletinBoard**: Basic UI in HTML analog REST-Schnittstellen, was Testing und Präsentation erleichtern wird

**4. Diverses**

- Der Virtual Host wurde bei den IT-Services bestellt und zwischenzeitlich auch geliefert.
- Gibt es Neuigkeiten betr. der Fehlermeldung in Tuple.java unter 1.8.0\_101 resp. 1.8.0\_102?
  - Update SNe: Auch mit Java 1.8.0\_111 ist der Fehler weiterhin vorhanden.
  - Seitens Herr Haenni gibt es noch keine Neuigkeiten diesbezüglich

**5. Weiteres Vorgehen**

- Nächste Treffen:
  - Dienstag, 8. November 2016 um 11:00 Uhr

Für das Protokoll

Verfasser: Sebastian Nellen, Biel, 25.10.16



Berner Fachhochschule  
BFH-TI

Sebastian Nellen  
Student  
Merkurstrasse 20  
3613 Steffisburg  
Telefon +41 79 737 57 52  
sebastian.nellen@students.bfh.ch

## Protokoll

Termin: Dienstag, 08. November 2016, 11:00 Uhr  
Ort: BFH, Rolex-Gebäude, Biel  
Anwesend: Haenni Rolf (Betreuer),  
Locher Philipp (Betreuer),  
Bürk Timo (Student),  
Nellen Sebastian (Student)

### Abstimmung Bachelor-Thesis

#### Traktanden

1. Gespräch mit Experte
2. Stand der Dinge
3. Fragen zur Implementation, Output aus Diskussion/Demo
4. Diverses
5. Weiteres Vorgehen

#### 1. Gespräch mit Experte

Das Treffen mit dem Experten fand am Dienstag, 01. November 2016 in Bern statt. Wir informierten Herrn Flueckiger generell über Thema e-Voting, über das Projekt 2 und zeigten die Planung des Thesis-Projekts (Systemdesign PoC, Ziele, Milestones, Use Cases).

Die Slides zum Meeting befinden sich im selben Verzeichnis wie dieses Protokoll.

#### 2. Stand der Dinge

- Demo des aktuellen Stands
- Wir sind aktuell ein wenig in Verzug, da die Einarbeitung und der Aufbau aufwändiger war als geplant. Trotz Mehraufwand konnte der Verzug leider (noch) nicht aufgeholt werden. Wir bleiben dran.

#### 3. Fragen zur Implementation, Output aus Diskussion/Demo

- Gibt es bereits Möglichkeiten zur Serialisierung und Deserialisierung von Unicrypt-Objekten?  
→ JA, sie heißen (sinngemäß) „convertToString, convertToBigInteger, ...“ und „convertFromstring, ...“
- Gibt es ein Referenz-Maven-Projekt mit Einbindung von Unicrypt, bei welchem wir die nötigen Daten übernehmen können?  
→ Eine Integration der aktuellen Unicrypt-Version ins Maven-Repository ist noch in Arbeit seitens EVG (S. Hauser), aktuell sollen wir Unicrypt noch „hard code“ verknüpfen
- Datenbank-Tabelle umbenannt: *tblParameters* anstatt *tblGenerators*  
→ Anpassung GET-Aufruf /generators zu /parameters notwendig

- Neue Datenbank-Tabelle: *tblCertificates*, welche E-Mail-Adressen zu Zertifikaten verbindet.  
→ Zukünftiger GET-Aufruf */certificates/{e-mail}* (Zert zu E-Mail) & */certificates* (alle)
- Anpassung der JsonData in der *tblVoters*-Tabelle, neu mit Mapping via E-Mail. Name, Vorname, und Ort wurde gestrichen, da die Informationen im Zertifikat ersichtlich sind.
- Anforderung/Tipp von Herr Locher: Alles was in die DB geschrieben wird, über die Funktion *mysqli::real\_escape\_string()* schreiben lassen.

#### 4. Diverses

- Der Fehler in *Tuple.java* unter Java 1.8.0\_101+ wurde durch Herr Haenni bereinigt und ins Unicrypt Git Repo gepushed.

#### 5. Weiteres Vorgehen

- Nächste Treffen:
  - o Dienstag, 22. November 2016 um 11:00 Uhr

Für das Protokoll

Verfasser: Sebastian Nellen. Biel, 08.11.16



Berner Fachhochschule  
BFH-TI

Sebastian Nellen  
Student  
Merkurstrasse 20  
3613 Steffisburg  
Telefon +41 79 737 57 52  
sebastian.nellen@students.bfh.ch

## Protokoll

Termin: Dienstag, 22. November 2016, 11:00 Uhr  
Ort: BFH, Rolex-Gebäude, Biel  
Anwesend: Haenni Rolf (Betreuer),  
Locher Philipp (Betreuer),  
Bürk Timo (Student),  
Nellen Sebastian (Student)

### Abstimmung Bachelor-Thesis

#### Traktanden

1. Stand der Dinge
2. Fragen zur Implementation, Output aus Diskussion/Demo
3. Weiteres Vorgehen

#### 1. Stand der Dinge

- Demo des aktuellen Stands
  - Eine Wahl kann erstellt werden, die entsprechenden Daten werden via Bulletin-Board in der Datenbank abgelegt
  - Ein Voter kann sich registrieren, die Private Credentials werden lokal abgelegt (aktuell noch in einem JSON-File) und die public Daten (Public Credential, Signatur, etc.) via Bulletin-Board in der Datenbank abgelegt
  - Ein Voter kann an einer Wahl teilnehmen, allerdings wird das Ballot noch nicht an das Bulletin-Board übertragen (work in progress)

#### 2. Fragen zur Implementation, Output aus Diskussion/Demo

- Haben Sie uns einen Tipp, wie wir jeweils vor dem Build der Apps automatisiert einen Build des util-Packages anstoßen können.  
→ Build on Save!
- Tipp von Herr Locher: Alles was in die DB geschrieben wird, über die Funktion `mysqli::real_escape_string()` schreiben lassen.  
→ Im Tutorial des SLIM-Frameworks wurde die Nutzung von PDO als Datenbank-Connector und die Nutzung von prepared statements empfohlen, weshalb wir auf `mysqli` verzichtet haben.  
→ Gemäss <http://stackoverflow.com/a/14012675> ist die Funktion aber mit PDO und prepared statements nicht notwendig.

#### 3. Weiteres Vorgehen

- Nächste Treffen:
  - Dienstag, 06. Dezember 2016 um 13:00 Uhr

Für das Protokoll

Verfasser: Sebastian Nellen. Biel, 22.11.16



Berner Fachhochschule  
BFH-TI

Sebastian Nellen  
Student  
Merkurstrasse 20  
3613 Steffisburg  
Telefon +41 79 737 57 52  
sebastian.nellen@students.bfh.ch

## Protokoll

Termin: Dienstag, 06. Dezember 2016, 13:00 Uhr  
Ort: BFH, Rolex-Gebäude, Biel  
Anwesend: Haenni Rolf (Betreuer),  
Locher Philipp (Betreuer),  
Bürk Timo (Student),  
Nellen Sebastian (Student)

### Abstimmung Bachelor-Thesis

#### Traktanden

1. Stand der Dinge
2. Fragen zur Implementation, Output aus Diskussion/Demo
3. Besprechung abzugebende Arbeit
4. Weiteres Vorgehen

#### 1. Stand der Dinge

- Demo des aktuellen Stands
  - Eine Wahl kann erstellt werden, die entsprechenden Daten werden via Bulletin-Board in der Datenbank abgelegt
  - Ein Voter kann sich registrieren, die Private Credentials werden lokal im Java-KeyStore abgelegt und die Public Daten (Public Credential, etc.) via Bulletin-Board in der Datenbank abgelegt
  - Ein Voter kann an einer Wahl teilnehmen
  - Ein Verifier kann eine Wahl überprüfen und auszählen. Das Resultat kann anschliessend an das Bulletin Board gesendet werden.

#### 2. Fragen zur Implementation, Output aus Diskussion/Demo

- Die nächsten zwei Wochen werden wir einerseits den Code überarbeiten, dokumentieren und auf der anderen Seite die Recherchen zum Thema Tor weiterführen.
  - Da Proof of Concept: eher Service verwenden als stundenlang mit Java-Implementation kämpfen
  - Mail an IT-Services: Firewall öffnen lassen, damit Tor-Test möglich warden.
- Optimierung Alphabet, um Platz zu sparen auf der DB.
  - Byte-Array (toByteArray() statt toString()) erzeugen und als Base64 ablegen
  - ZIP und dann als Blob ablegen
- Visualisieren, welche Schritte beim Verifier durchlaufen werden.
- Fokus:
  - Input-Validierung sparen (und erwähnen)
  - Testing trotzdem notwendig
  - KeyStore- und Passwort-Thematik
    - Zertifikat im OS Keystore, App muss Erlaubnis anfragen, Private Credentials über Public Key des Zertifikats verschlüsseln
    - max. 2 Passwörter
    - alpha & beta kann via Unicrypt gepaart werden

- Die Beweise sind aktuell noch unabhängig von der Stimme  $e$ . Diese können via ChallengeGenerator -> Challenge erstellen (Additional Input  $\leftarrow$  String (Stimme)), FiatShamir... abhängig von  $e$  gemacht werden.
- Unicrypt (2.2) neu im Maven-Repo, kann integriert werden
- Tipps für die Verteidigung resp. Präsentation:
  - Antwort auf Frage „Warum müssen Stimmen nicht verschlüsselt sein?“
  - Bereit sein für kleine Detail-Fragen
  - Demo-Attacken:
    - „Stimme *Nein* in ein *Ja* ändern“
    - Beweise attackieren (Achtung geben, an welcher Zahl wir schrauben, eher  $Z_p^*$ )
  - Folie mit riesigen Zahlen ( $x^y \bmod q$ ), damit die Leute das Problem sehen
  - Vote via Tor oder ohne – als Option → BFH-IP oder nicht

### 3. Besprechung abzugebende Arbeit

- In welcher Form wir die Arbeit erwartet?
- Zusammenfassung Projekt 2
  - Zusammenfassung Protokoll
  - Warum Everlasting Privacy
- Beschreibung Produkt

### 4. Weiteres Vorgehen

- Nächstes Treffen:
  - Dienstag, 10. Januar 2017 um 10:00 Uhr
  - Bei Fragen betr. Unicrypt/Implementation wenden wir uns per Mail an Herrn Locher (vorzugsweise KW50)
  - Wir versenden noch einen Terminvorschlag an Herr Haenni für KW51. Besprechungsthema: abzugebende Arbeit, deren Struktur und Inhalt

Für das Protokoll

Verfasser: Sebastian Nellen, Biel, 06.12.16



Berner Fachhochschule  
BFH-TI

Sebastian Nellen  
Student  
Merkurstrasse 20  
3613 Steffisburg  
Telefon +41 79 737 57 52  
sebastian.nellen@students.bfh.ch

## Protokoll

Termin: Dienstag, 10. Januar 2017, 10:00 Uhr  
Ort: BFH, Rolex-Gebäude, Biel  
Anwesend: Haenni Rolf (Betreuer),  
Locher Philipp (Betreuer),  
Bürk Timo (Student),  
Nellen Sebastian (Student)

### Abstimmung Bachelor-Thesis

#### Traktanden

1. Dokumentation
2. Offene Pendenzien
3. Testing
4. Stand Prototyp

#### 1. Dokumentation

- Feedback der Betreuer über «Dokumentation v0.1»?
  - Schriftgrösse auf 11pt setzen
  - Einleitung ist zu kurz, wirkt minimal und knapp – sollte für jedermann lesbar sein
    - Konzept, Beitrag der Arbeit (Hauptresultat), Teaser
    - Zusammenfassung vorab
    - Überblick geben, was in der Arbeit in geschrieben steht
    - Stand der Dinge (Situation CH)
  - Unterkapitel in Einleitung
  - Management Summary auf Deutsch, Abstract auf English (präziser)
  - Dokument wirkt überstrukturiert, lieber weniger Subsections machen
  - S. 9 „Es besteht aber trotzdem ein Risiko, dass ein Wählerprofil erstellt und auf einen Wähler (oder eine Gruppe von Wählern) geschlossen werden kann.“ → Mehr schreiben, oder weglassen
  - Paragraphen ja, kein Zeilenvorschub (keine «\\» im Latex-Fliesstext)
  - 4.2.4. Bulletin-Board -> simple ändern
  - JWS: sicher stellen, dass genau beschrieben ist, wo JWS verwendet wird
  - Bei Lückenfüllern z.B. zw. 2 und 2.1: Ausblick, um was es im Kapitel geht
  - Besser den Faden aufnehmen anstatt zusammenfassen, was bereits im vorherigen Kapitel geschrieben wurde
  - Aufpassen mit Formulierung „Man muss darauf vertrauen, dass ...“ wenn etwas es verifizierbar ist
  - Annahmen aufführen und anschliessend nicht mehr behandeln (z.B. Unsecure Hardware)
- Welche Lieferobjekte sind in welcher Form gewünscht?
  - In welcher Form möchten die Betreuer die Dokumentation erhalten? Digital oder in Papierform?
    - Gedrucktes Exemplar für Betreuer am Finaltag

- PDF am 23:59 Uhr am Do, 19.12.2016
- Exemplar für Archiv
  - Dokumentation gedruckt
  - Source Code (Export von GIT) auf CD oder USB-Stick
- Exemplar für Experte: PDF per Mail
- In welcher Form sollen wir die «Installationsanleitung» in die Dokumentation aufnehmen?
  - Entweder in ein Kapitel (dann aber sauber) oder im Anhang
- 2. **Offene Pendenden**
  - In Ausblick aufnehmen: Optimierung Alphabet, um Platz zu sparen auf der DB.
    - Byte-Array (toByteArray) statt toString() erzeugen und als Base64 ablegen
    - ZIP und dann als Blob ablegen
  - Unicrypt (2.2) neu im Maven-Repo, muss noch integriert werden
  - Die KeyStore-Thematik mussten wir depriorisieren und uns um die Dokumentation kümmern.
- 3. **Testing**
  - Sind wir auf der richtigen Spur? -> Demo JUnit und PHPUnit -> Ja
  - Werden noch weitere Tests benötigt?
    - Idee: End-to-End -> Element in DB und wieder holen - immer noch OK
    - Zeigen der Testkonzepte: Unit, Integration (GUI), End-to-End
    - Detaillierungsgrad ist weniger wichtig, aber zeigen, dass wir an alles gedacht haben und wie
- 4. **Stand Prototyp**
  - Visualisierung der Auszählschritte beim VerifierApp implementiert -> Demo
    - Tabelle mit Daten machen
    - PRÄSENTATION -> zuerst erklären, dann klicken | Fokus auf's Wesentliche
    - Weniger Daten! (statt 7x Ja und gültig)
    - Sprechende Bezeichnungen, z.B. statt U\_hat > election credential
  - Stimmabgabe über Tor implementiert. -> Demo
    - Wir verwenden den Proxy-Service des Tor Projekts.
    - Standardmäßig wird jeweils nur alle ca. 10 Minuten ein neuer Tor-Circuit gewählt und aufgebaut. Dies ist aktuell immer noch der Fall.
    - Per default: Use Tor (IF-Check einbauen: Läuft Tor? - Wenn Tor nicht läuft, disable Checkbox)
    - IP in Datenbank abspeichern (ggf. mit Lookup aus welchem Land per Knopfdruck)
  - Die Beweise sind nun abhängig von der Stimme e.
  - UI wird nach Abschluss Dokumentation noch optimiert.

Für das Protokoll

Verfasser: Sebastian Nellen. Biel, 10.01.17