Elektronische Abstimmungen mit homomorpher Verschlüsselung

Bachelor Thesis

Martin Bächtold (baecm2@bfh.ch)
Christian Jauslin (jausc1@bfh.ch)
Jürg Ritter (rittj1@bfh.ch)

Betreut durch
Prof. Dr. Rolf Haenni
Prof. Reto Koenig

Experte
Prof. Dr. Andreas Spichiger

Berner Fachhochschule - Technik und Informatik
Januar 2011

Erklärung der Diplomanden

Selbständige Arbeit

Wir bestätigen mit unseren Unterschriften, dass wir unsere Bachelor Thesis selbständig durchgeführt haben. Alle Informationsquellen (Fachliteratur, Besprechungen mit Fachleuten, usw.), die wesentlich zu unserer Arbeit beigetragen haben, sind in unserem Arbeitsbericht im Anhang vollständig aufgeführt.

Bern, 21. Januar 2011

Martin Bächtold

Christian Jauslin

Jürg Ritter

Abstract

Elektronische Abstimmungen übers Internet durchzuführen, ist aus Sicht der IT-Security ein äusserst schwieriges Unterfangen. Es geht darum, die Wähler beim Wahlvorgang eindeutig zu identifizieren, ohne dabei das Stimmgeheimnis zu verletzen. Mit entsprechenden kryptographischen Mitteln kann dies aber garantiert werden.

Aufgabe

Ziel dieser Bachelor-Thesis ist ein E-Voting-System, welches als Proof-of-Concept für die Anwendbarkeit dieser kryptographisch anspruchsvollen Methoden in der Praxis dient, wobei durch die verwendeten kryptogaphischen Techniken die Bedingungen aus Sicht der IT-Security gewährleistet werden können.

Lösung

Die implementierte Lösung besteht aus einem Web-Interface, das es einem Wähler ermöglicht an einer Abstimmung teilzunehmen, das Resultat und alle beteiligten Parameter einzusehen, und so die Korrektheit des Abstimmungsprozesses zu überprüfen. Zudem wurde ein Java-Client programmiert, mit welchem Aufgaben ausgeführt werden können, die vor Eröffnung der Abstimmung nötig sind, sowie für die Entschlüsselung des Abstimmungsresultates.

Inhaltsverzeichnis

1. Einleitung

Das vorliegende Dokument ist im Rahmen des Abschlussprojekts einer Bachelor-Thesis an der Berner Fachhochschule, Technik und Informatik entstanden. Die nachfolgende offizielle Aufgabenstellung der Bachelor-Thesis[1] ist eine prägnante Beschreibung, worüber es sich in dieser Arbeit handelt:

"Elektronische Abstimmungen übers Internet durchzuführen, ist aus der IT-Security Perspektive ein äusserst schwieriges Unterfangen. Es geht darum, die Wähler beim Wahlvorgang eindeutig zu identifizieren, ohne dabei das Stimmgeheimnis zu verletzen. Mit entsprechenden kryptographischen Mitteln kann dies aber garantiert werden. Ein vielversprechender Ansatz in der Literatur basiert auf sogenannten "homomorphen Verschlüsselungen" und "Zero-Knowledge Beweisen". Diese Techniken wurden in einer vorgängigen Projektarbeit bereits studiert und ansatzweise implementiert. Das Ziel dieser Bachelor-Arbeit besteht darin, mit Hilfe der oben genannten Techniken ein voll funktionstüchtiges E-Voting System zu implementieren. Dazu braucht es verschiedene Komponenten einer Server-Client-Architektur, die miteinander kommunizieren und einzeln realisiert werden müssen. Diese Komponenten sowie entsprechende Schnittstellen zu den anderen Komponenten müssen im Detail spezifiziert und implementiert werden. Es geht dabei darum, ein Proof-of-Concept zu realisieren, um die Anwendbarkeit von solch kryptographisch anspruchsvollen Methoden in der Praxis besser beurteilen zu können."

1.1. Organisation

Diese Bachelor-Thesis wurde als Gruppenarbeit von drei Bachelor Studenten der Berner Fachhochschule, Technik und Informatik, mit dem Schwerpunkt "IT-Security" realisiert. Betreut wurden sie von zwei Dozenten, die sich im Rahmen der Forschungstätigkeit der Schule mit dem Thema E-Voting befassen.

[1] https://www2.ti.bfh.ch/fbi/2011/Studienbetrieb/BaThesisHS10/aufgabestellungen/IHNR1-1-10-de.xml

Studenten

- Martin Bächtold
- Christian Jauslin
- Jürg Ritter

Betreuer

- Prof. Dr. Rolf Haenni
- Prof. Reto Koenig

Experte

- Prof. Dr. Andreas Spichiger

1.2. Vorarbeiten

Im Rahmen des Moduls 7302 “Projekt 2” wurden von den drei Studenten die theoretischen Grundlagen für die Bachelor-Thesis erarbeitet und in Form einer schriftlichen Arbeit, einer “Krypto-Library” in Java und einer Präsentation erfolgreich abgeschlossen. Die Bachelor Thesis ist eine Weiterführung des Moduls “Projekt 2” und besteht hauptsächlich aus der Implementation einer Software-Applikation zum gewählten Thema.

1.3. Aufbau des vorliegenden Dokuments

Im ersten Teil bietet dieses Dokument einen Überblick zum Thema E-Voting und es werden grundlegende Konzepte erklärt. Anschliessend wird genauer auf die Theorie des CGS97 [1] Protokolls, welches die Basis für die Umsetzung dieses Projekts bietet, eingegangen. Das im Verlaufe dieser Abschlussarbeit entstandene Software-Produkt wird dann im letzten Teil im Detail erläutert.

2. Einführung zu E-Voting

Der Prozess einer herkömmlichen Abstimmung ist auf den ersten Blick unspektakulär. Eine stimmberechtigte Person gibt eine Stimme in einem Wahllokal oder per Briefpost ab, die eingegangenen Stimmen werden ausgezählt und das Resultat wird publik gemacht. Man könnte meinen, dieser Prozess wäre mit geringem Aufwand in die digitale Welt übertragbar. Ganz unscheinbar laufen im Hintergrund jedoch Prozesse ab, die in der heutigen Informatik nicht auf triviale Art und Weise abbildbar sind. Hier ein paar Beispiele:

- Sicherstellen dass jeder Wahlberechtigte nur einmal abstimmen kann (z.B. Wahlausweise)
- Anonymisierung der Stimmen (Stimmzettel und Wahlausweise werden in separate Urnen geworfen)
- Stimmzettel dürfen nicht verändert werden können (Wahlurnen sind plombiert)

Ausserdem ist selbst in der realen Welt die Transparenz einer solchen Abstimmung mangelhaft, als Wähler kann man nicht davon überzeugt sein, dass die eigene Stimme korrekt in das Abstimmungsresultat einbezogen wurde.

2.1. Anforderung an ein E-Voting-System

Nebst den Anforderungen, die für andere informationstechnische Systeme gelten (zum Beispiel Schutzmassnahmen gegen Denial of Service-Attacken), müssen für E-Voting-Systeme formalere Kriterien eingehalten werden: [2]

- **Korrektheit:** Eingegangene Stimmen dürfen nicht verändert oder gelöscht werden, ungültige Stimmen dürfen nicht gezählt werden
- **Demokratisch:** Nur berechtigte Wähler dürfen an der Abstimmung teilnehmen. Ausserdem muss sichergestellt werden, dass ein berechtigter Wähler nur eine Stimme pro Abstimmung abgeben darf

- **Vertraulichkeit:** Das System muss Anonymität bieten, d.h. niemand (auch nicht die Wahlbehörde) soll feststellen können, für welche Option ein Wähler gestimmt hat
- **Überprüfbarkeit:** Jeder Wähler kann selbständig überprüfen, ob die eigene Stimme auch korrekt gezählt worden ist
- **Gerechtigkeit:** Das E-Voting-System darf keine Zwischenresultate zugänglich machen bevor die Abgabefrist der Stimmen abläuft

2.2. E-Voting-Systeme in der Praxis

E-Voting-Systeme sind zum heutigen Zeitpunkt noch nicht weit verbreitet. Erst einige Schweizer Kantone haben E-Voting-Systeme in einem Pilotversuch eingeführt und damit bereits Abstimmungen auf kommunaler oder kantonaler Ebene durchgeführt. Leider zielen diese Systeme mehrheitlich auf die Vereinfachung und Automatisierung des Abstimmungsprozesses und somit der Reduktion von Kosten ab und es mangelt an Transparenz. Ein elektronischer Wahlzettel wird an dieses System gesendet, was aber anschliessend mit diesem Stimmzettel passiert ist für einen Wähler ohne Hintergrundwissen nicht nachvollziehbar. Die oben genannten Anforderungen an ein E-Voting-System werden somit nur teilweise oder gar nicht erfüllt. Genau hier möchten neuartige Konzepte ansetzen und versuchen, mit Hilfe von moderner Kryptographie diesen Mangel an Transparenz zu beheben.

2.3. E-Voting mit homomorpher Verschlüsselung (CGS97)

Eine Idee um die Probleme der vorhandenen E-Voting-Systeme zu beheben, haben die Kryptographen Ronald Cramer, Rosario Gennaro und Berry Schoenmakers im Rahmen der Eurocrypt Konferenz 1997 vorgeschlagen und dafür ein E-Voting-Protokoll vorgestellt. [1] Das Protokoll weist folgende Eigenschaften auf:

- Nur berechtigte Personen dürfen eine Stimme abgeben
- Das Endergebnis ist universell überprüfbar, Stimmen werden korrekt ausgezählt und ungültige Stimmen ignoriert
- Jede Stimme ist geheim, solange die Anzahl konspirierender Behördenmitglieder einen bestimmten Schwellwert nicht überschreitet

- Fehlerhaftes Verhalten kann toleriert werden, keine Koalition von Wählern beliebiger Grösse kann die Wahl stören
- Betrügerische Wähler werden erkannt und von der Wahl ausgeschlossen
- Niemand kann die Stimme eines Wählers kopieren (auch nicht unabsichtlich)
- Keine Interaktion zwischen den Wählern nötig

Das System ist aber nicht quittungsfrei und somit nicht erpressungsresistent[1]. Das heisst, ein Wähler könnte einer Drittperson beweisen, ob oder welche Stimme er abgegeben hat. Das E-Voting-Protokoll von Ronald Cramer et al. [1] nutzt das ElGamal-Verschlüsselungsverfahren, wobei ein Wähler seine Stimme mit einem öffentlich zugänglichen Schlüssel verschlüsselt. Der Wähler schickt diese verschlüsselte Stimme zusammen mit einem Beweis, dass es sich um eine gültige Stimme handelt (zum Beispiel entweder eine *Ja*- oder *Nein*-Stimme)[2], an ein öffentliches schwarzes Brett[3] in einem nur für ihn vorgesehenen Bereich. Zudem darf es niemandem möglich sein, die abgelegten Stimmen vom schwarzen Brett zu entfernen. Ausserdem müssen Denial-Of-Service-Attacken gegen die Server des schwarzen Brettes ausgeschlossen werden. Die Implementation und die Techniken zur Absicherung des schwarzen Brettes sind nicht Gegenstand dieses Projekts. Der für die Entschlüsselung des Abstimmungsergebnisses benötigte Schlüssel wird in diesem Protokoll auf mehrere Behördenmitglieder[4], sogenannte Trustees aufgeteilt. Dank einem Schwellwert-Verfahren[5] müssen eine bestimmte Anzahl Behördenmitglieder ihre Teilschlüssel einbringen, um das Resultat zu entschlüsseln. Dabei sind aber nicht alle Teilschlüssel nötig. Dies bietet einen gewissen Schutz vor korrupten oder unabsichtlich falsch agierenden Behördenmitglieder.

2.3.1. Umsetzung von CGS97 durch das HELIOS Projekt

HELIOS ist eine unabhängige Nonprofit-Organisation, welche das gleichnamige webbasierte "Open-Audit"-Votingsystem entwickelt. Der Quellcode der Client- und Serverkomponenten ist frei verfügbar und auf `http://heliosvoting.org` können kostenlos Abstimmungen durchgeführt werden.

[1] engl. coercion-resistent
[2] engl. ballot = Verschlüsselte Stimme + Beweis
[3] engl. bulletin board
[4] engl. authorities, trustees
[5] engl. threshold

Abbildung 2.1.: Einstiegsseite Helios

Neben der Durchführung und Überprüfbarkeit von Abstimmungen liegt der Fokus von HELIOS vor allem in der Steigerung der Akzeptanz von elektronischen Wahlen in der breiten Öffentlichkeit.

Wie die vorliegende Bachelorarbeit basiert auch HELIOS auf den Abläufen und kryptographischen Grundlagen des CGS97 Protokolls. [1]

3. Kryptographische Grundlagen

Die Implementation eines E-Voting-Systems gemäss dem CGS97 Protokoll [1] basiert hauptsächlich auf drei kryptographischen Bausteinen. Dazu gehören das asymmetrische Kryptosystem ElGamal, ein Threshold Secretsharing Verfahren und Zero-Knowledge-Beweise. Diese Bausteine werden in diesem Abschnitt erläutert.

3.1. ElGamal

Der ElGamal Kryptoalgorithmus ist ein asymmetrisches Kryptosystem, welches sich das mathematische Problem des diskreten Logarithmus in finiten Zahlengruppen zu Nutze macht. Es wurde 1985 vom Ägypter Taher Elgamal entwickelt. ElGamal arbeitet mit multiplikativ modularen Gruppen.

3.1.1. Domainparameter

Als Domainparameter werden drei Grössen gebraucht, wir werden diese p, q, und g nennen.

Der Parameter p sei eine Primzahl und definiere die endliche multiplikativ modulare Zahlengruppe $\mathbb{Z}_p^*$. Diese Gruppe enthält alle Elemente von 1 bis $p-1$ und hat in Folge dessen die Mächtigkeit $|\mathbb{Z}_p^*| = p-1$.

Der Parameter q definiert die Mächtigkeit einer möglichst grossen Untergruppe G_q von $\mathbb{Z}_p^*$. Eine Untergruppe impliziert, dass alle Elemente von G_q auch in $\mathbb{Z}_p^*$ vorhanden sind. q muss ein Teiler von $p-1$ sein und ist idealerweise möglichst gross. In der Praxis ist es oft einfacher, zuerst q zu definieren und danach p mit Hilfe eines Faktors k nach folgendem Schema zu finden:

$$p = kq + 1$$

Danach muss noch getestet werden, ob man mit p ebenfalls eine Primzahl gefunden hat. Ist dies nicht der Fall, muss q neu definiert und das ganze solange wiederholt werden, bis p und q beide prim sind.

Ein Generator g ist ein Element der Gruppe $\mathbb{Z}_p^*$ und generiert die Gruppe G_q. Der Generator muss die Eigenschaft besitzen, dass er bei der Potenzierung mit allen Elementen

der Gruppe $\mathbb{Z}_q$ alle Elemente der Gruppe G_q generiert. Dank des Problems des diskreten Logarithmus in finiten Zahlengruppen haben wir so eine irreversible Funktion, die alle Elemente in $\mathbb{Z}_q$ in ein anderes Element in G_q abbilden kann. Ein Beispiel soll dies veranschaulichen:

Sei $q = 11$, $g = 2$ und $p = 23$.

i	1	2	3	4	5	6	7	8	9	10
$g^i \pmod p$	2	4	8	5	10	9	7	3	6	1

Hier sieht man, dass jedes Element von $\mathbb{Z}_q$ wieder als Resultat von $g^i \pmod p$ erscheint, deshalb ist 2 ein Generator von G_{11}. Nicht alle Elemente von $\mathbb{Z}_{11}$ sind aber Generatoren wie das folgende Beispiel mit $g = 9$ zeigt:

i	1	2	3	4	5	6	7	8	9	10
$g^i \pmod p$	9	4	3	5	1	9	4	3	5	1

Bei diesem Beispiel stellt man fest, dass nur ein paar Elemente aus $\mathbb{Z}_{11}$ in den Resultaten auftauchen und dass sich diese Elemente immer in der gleichen Reihenfolge wiederholen. Generatoren von G_{11} sind demzufolge 2, 6, 7 und 8.

Im Allgemeinen ist es nicht einfach zu überprüfen, ob eine Zahl ein Generator $\pmod p$ ist. Kennt man jedoch die Primfaktoren von $p - 1$ kann man dies recht einfach feststellen. Will man wissen ob ein Wert g ein Generator $\pmod p$ ist, zerlegt man $p - 1$ in seine Primfaktoren q_i und berechnet

$$g^{(p-1)/q_i} \pmod p$$

für alle Primfaktoren q_i. Nimmt das Resultat für mindestens einen Primfaktor q_i den Wert 1 an, so ist g **kein** Generator. Zur Veranschaulichung machen wir ein Beispiel mit $p = 11$ und untersuchen ob $g = 2$ und $g = 3$ Generatoren $\pmod p$ sind.

Die Primfaktoren von $(p - 1)$ sind 2 und 5. Um die beiden Kandidaten zu überprüfen rechnen wir nun

$$2^{10/5} \pmod{11} = 4$$
$$2^{10/2} \pmod{11} = 10$$

Die Resultate sind in beiden Fällen ungleich 1, somit ist 2 ein Generator $\pmod{11}$. Im Fall von $g = 3$ rechnen wir

$$3^{10/5} \pmod{11} = 9$$
$$3^{10/2} \pmod{11} = 1$$

Hier nimmt das Resultat in einem Fall den Wert 1 an, somit ist 3 kein Generator $(\text{mod } 11)$. [3]

3.1.2. Verschlüsselung und Entschlüsselung

Wie jedes asymmetrische Kryptoverfahren braucht auch ElGamal einen Public Key um Nachrichten zu verschlüsseln, sowie einen Private Key um diese wieder zu entschlüsseln. Sind die Domainparameter p, q und g erfolgreich definiert, geschieht die Erstellung dieser beiden Keys folgendermassen:

- Private Key $s \in_R \mathbb{Z}_q$
- Public Key $h = g^s \pmod p$

Verschlüsselung

Um eine Nachricht m zu verschlüsseln, wird ein modifizierter ElGamal angewendet. Man benutzt dazu einen zusätzlichen, zufällig gewählten Parameter $k \in \mathbb{Z}_q$. Dieser hat den Effekt, dass bei jeder Verschlüsselung des identischen Plaintexts nicht immer derselbe Ciphertext entsteht. Die Verschlüsselung geschieht nach folgendem Schema:

- Sei $k \in_R \mathbb{Z}_q$
- $E(m) = (x, y) = (g^k \pmod p, h^k m \pmod p)$

Das Tupel (x, y) ist nun der Ciphertext vom Plaintext m.

Entschlüsselung

Der Ciphertext (x, y) kann nun mit dem Private Key s wie folgt entschlüsselt werden:

$$m = y \cdot x^{p-1-s} \pmod p$$

Somit wurde der ursprüngliche Plaintext m wiederhergestellt.

3.2. ElGamal und Homomorphismus

Das Kryptosystem ElGamal hat eine Eigenschaft, die es erlaubt, mit Ciphertexten zu rechnen. Dies können wir in einem E-Voting-System einsetzen. Wir möchten dabei die Summe aller verschlüsselten Stimmen bilden und dann nur das Resultat entschlüsseln. Somit müssen die verschlüsselten Stimmen nie entschlüsselt werden, um das definitive

Endresultat einer Abstimmung sichtbar zu machen. Um diese Eigenschaft nutzen zu können, müssen wir jedoch eine kleine Änderung am ElGamal Verschlüsselungsschema machen, denn ElGamal ist multiplikativ homomorph:

$$E_h(v_1) \cdot E_h(v_2) = E_h(v_1 \cdot v_2)$$

E sei hier die ElGamal Verschlüsselungsfunktion und h der Public Key der verwendet wurde. Um nicht das Produkt der Stimmen, sondern deren Summe zu erhalten, wird der zu verschlüsselnde Wert exponiert. Somit werden die Exponenten bei der Multiplikation bei gleichbleibender Basis addiert:

$$g^{v_1} \cdot g^{v_2} = g^{(v_1+v_2)}$$

Die ElGamal Verschlüsselungs- und Entschlüsselungsfunktion müssen leicht modifiziert werden, damit diese Eigenschaft genutzt werden kann. Die Verschlüsselungsfunktion sieht dann so aus:

$$(x, y) = (g^k \bmod p, g^x h^k \bmod p)$$

Wenn wir die aus dieser Verschlüsselungsfunktion entstehenden Ciphertexte der Stimmen $v_1, \ldots, v_n$ miteinander multiplizieren und danach mit der Entschlüsselungsfunktion entschlüsseln, erhalten wir ein Resultat in der Form

$$\prod_{i=1}^{n} g^{v_i} \bmod p = g^{\sum_{i=1}^{n} v_i} \bmod p$$

Um $\sum_{i=1}^{n} v_i$ zu erhalten, müsste der diskrete Logarithmus in einer endlichen Gruppe berechnet werden, was nicht effizient lösbar ist. Da jedoch die Menge aller möglichen Resultate bekannt ist, wendet man die "Brute Force"-Methode an. Im schlimmsten Fall sind so viele Versuche nötig, wie es Stimmen gibt.

$$g^{1,\ldots,i} \bmod p \stackrel{?}{=} g^{\sum_{i=1}^{n} v_i} \bmod p$$

Stimmt unser Versuch mit $g^{\sum_{i=1}^{n} v_i}$ überein haben wir mit i die Summe der verschlüsselten Stimmen gefunden.

3.3. Secret Sharing und Threshold

Ein grundsätzliches Problem beim Einsatz von Kryptographie in der Praxis ist, dass die zur Verschlüsselung benutzten Private Keys selbst auch wieder verschlüsselt werden

müssen, sobald die (ersten) Private Keys an einem physikalisch unsicheren Ort gespeichert werden. Dieser Endloszyklus kann durchbrochen werden, wenn ein oder mehrere Private Keys an einem physikalisch sicheren Ort gespeichert werden (z.B. auf einem Hardware Security Module *HSM*).

Als Beispiel sei der Private Key einer Certification Authority *CA* erwähnt, welcher dafür benutzt wird, Zertifikate auszustellen. Es wäre verheerend, falls so ein Private Keys in falschen Hände geriete oder verloren gegangen würde.

Es liegt auf der Hand, dass ein solcher Private Key gleichzeitig *geheim* gehalten werden, aber trotzdem *verfügbar* sein muss. Man ist geneigt zu denken, dass der Private Key an nur einem geheimen Ort gespeichert werden kann, damit er nicht Unbefugten in die Finger gerät, und andererseits an möglichst vielen Orten abgelegt ist, damit er hoch verfügbar ist. Secret Sharing will dieses Problem lösen.

Einfach gesagt geht es darum, einen Private Key[1] s in n Teilschlüssel aufzuteilen, wobei nicht alle Teilschlüssel notwendig sind, um den Private Key wiederherstellen zu können.

3.3.1. Secret Sharing mit Shamir's Threshold-Verfahren

Dieses Verfahren wurde von Adi Shamir 1979 [4] entwickelt, welcher auch die nachfolgende Implementierung vorgeschlagen hat. Wenn wir ein solches Schema haben, können wir wie oben erwähnt einen Private Key aufteilen und die n Teile an unterschiedlichen Orten aufbewahren. Um den Private Key wiederherzustellen, benötigt man mindestens t Teile, d.h. solange man nicht mehr als $n - t$ Teile verliert, kann man den Private Key immer noch wiederherstellen.

Schlüsselverteilung

Um ein Secret $s \in \mathbb{Z}_q$ über n Teilnehmer zu verteilen, wählen wir ein Polynom $f(x)$ mit Grad $t-1$ mit zufälligen Koeffizienten a_i, $0 \leq i \leq t-1$, aus $\mathbb{Z}_q$, so dass $f(0) = s$ ist.

$$f(x) = s + \sum_{i=1}^{t-1} a_i x^i = s + a_1 x^1 + a_2 x^2 + ... + a_{t-1} x^{t-1}$$

Jeder der n Teilnehmer erhält nun einen Teilschlüssel s_j, also einen Stützpunkt $(j, f(j))$ auf diesem Polynom. Die einzige Ausnahme ist der Stützpunkt $(0, f(0))$, welcher dem Secret s entspricht und nicht verteilt werden darf.

[1] Man spricht allgemein von einem *Secret*, also einem allgemeinen Geheimnis

Schlüsselwiederherstellung

Wenn nun beliebige t von n Teilnehmer kooperieren, ist die Rekonstruktion des Polynoms $f(x)$ und somit des Private Keys $f(0) \rightarrow s$ mittels Lagrange Interpolation möglich.

$$\begin{aligned} f(x) &= \sum_{i=0}^{t-1} f(x_i)\lambda_i(x) \\ &= \sum_{i=0}^{t-1} f(x_i) \prod_{j=0, j\neq i}^{t-1} \frac{x - x_j}{x_i - x_j} \end{aligned}$$

Da uns nur $f(0)$ interessiert, können wir mit

$$s = f(0) = \sum_{i=0}^{t-1} f(x_i) \prod_{j=0, j\neq i}^{t-1} \frac{-x_j}{x_i - x_j}$$

das Secret s berechnen.

3.3.2. Verteilte Schlüsselgenerierung

Wenn eine einzelne Person den Private Key erstellt und an verschiedene Personen verteilt, ist ihr ein (zu) grosses Vertrauen entgegenzubringen. Vor allem für E-Voting-Syteme ist es wünschenswert, dass auf eine solche vertrauenswürdige dritte Partei verzichtet werden kann. Die Lösung ist ein Verfahren, welches einen geheimen Private Key so aufteilt, dass einzelne an der Schlüsselerzeugung beteiligte Personen nichts über den Private Key aussagen können. Torben Pedersen [5] beschreibt dieses Verfahren in folgenden Schritten:

- Jeder Teilnehmer $A_j, 1 \leq j \leq n$, veröffentlicht $\beta_j = g^{\alpha_j}, \alpha_j \in_R \mathbb{Z}_q$, und beweist mittels eines Zero-Knowledge-Beweises (siehe 3.4), dass er im Besitz von α_j ist.
- Jeder Teilnehmer A_j wählt ein Polynom $f_j(x)$ mit zufälligen Koeffizienten a_j aus $\mathbb{Z}_q$ mit Grad $t-1$, so dass $f_j(0) = s_j = \alpha_j$ ist.
- Jeder A_j veröffentlicht die Werte $g^{a_{ji}}, 1 \leq i \leq t-1$, ohne $g^{a_{j0}}$, da dieser Wert bereits im ersten Schritt mit $\beta_j = g^{\alpha_j} = g^{a_{j0}}$ veröffentlicht wurde.
- Jeder A_j sendet signiert und verschlüsselt den Wert $f_j(i), 1 \leq i \leq n$, an den Teilnehmer A_i, für $i = 1 \leq i \leq n, i \neq j$.
- Jeder A_i verifiziert die Signatur und überprüft den erhaltenen Wert $f_j(i)$ mit

$$g^{f_j(i)} = \prod_{l=0}^{t-1} g^{a_{jl} \cdot l \cdot i}$$

Falls die Verifizierung nicht gültig ist, wird ein *Complaint*[2] gegen den betroffenen Teilnehmer veröffentlicht.

- Falls nun mehr als $t-1$ Teilnehmer einen *Complaint* gegen A_j veröffentlicht haben, wird dieser disqualifiziert.

- Jeder A_j setzt nun seinen Stützwert s_j, indem er die Summe aller erhaltenen $f_i(j)$ bildet.

$$s_j = \sum_{i=1}^{n} f_i(j)$$

  Nun kann jeder Teilnehmer seinen zugehörigen Teil des Public Keys $h_j = g^{s_j}$ publizieren.

- Mit Hilfe der publizierten Werte $g^{a_{ji}}$ jedes qualifizierten Teilnehmers, können alle den Public Key berechnen, in dem sie $g^{a_{j0}}$ Multiplizieren.

$$h = \prod_{j=1}^{n} g^{a_{j0}}$$

- Wenn nun t oder mehr Teilnehmer A_j zusammenarbeiten kann das Secret s wie folgt berechnet werden:

$$s = \sum_{j=1}^{t} \lambda_j(0) \cdot s_j$$

$$\lambda_j(0) = \prod_{i=1, j \neq i}^{t} \frac{-x_i}{x_j - x_i}$$

3.3.3. Verteilte Entschlüsselung

Das Ziel ist nun, dass die Teilnehmer $A_j, 1 \leq j \leq n$ zusammen eine ElGamal-verschlüsselte Nachricht $(x, y) = (g^k, h^k \cdot m)$ mit $h = g^s = g^{\sum s_j}$ entschlüsseln können, ohne explizit den Private Key s konstruieren zu müssen. Der geheime Private Key s ist auf n Teilnehmer $A_j, 1 \leq j \leq n$, aufgeteilt.

- Jeder A_j publiziert $w_j = x^{s_j}$ und beweist mittels Zero-Knowledge-Beweis, dass $\log_g(h_j) = log_x(w_j)$ gilt.

- Wenn nun t oder mehr Personen den Zero-Knowledge-Beweis bestehen, kann die

[2] Complaint = Anzeige, Anklage

Nachricht $m = (x, y)$ wie folgt entschlüsselt werden.

$$m = (g^{-k})^s \cdot (g^s)^k \cdot m = (g^k)^{-\sum \lambda_j(0) \cdot s_j} \cdot h^k \cdot m$$

Vereinfachen wir mit $x = g^k$ und $y = h^k \cdot m$ ergibt das

$$m = x^{-\sum \lambda_j(0) \cdot s_j} \cdot y = \prod x^{-\lambda_j(0) \cdot s_j} \cdot y$$

Nun setzen w_j für x ein und erhalten

$$m = \prod w_j^{-\lambda_j(0)} \cdot y = \frac{y}{\prod w_j^{\lambda_j(0)}}$$

Die Entschlüsselung ist für alle Teilnehmer verifizierbar, da sich jeder mit Hilfe des Zero-Knowledge-Beweises überzeugen kann, dass richtig entschlüsselt wird.

3.4. Sigma-Protokolle

Beweissysteme dienen u.a. dazu, einen korrekten Protokollablauf zu gewährleisten. Dabei beweisen Teilnehmer anderen Teilnehmern im Protokollablauf, dass sie das Protokoll korrekt befolgt haben. In einem E-Voting-System kann beispielsweise ein Wähler beweisen, dass er eine korrekte Wahloption verschlüsselt hat (bei einer *Ja/Nein-Wahl* sind z.B. nur die Optionen 0 oder 1 zulässig). Wir sprechen im weiteren von Prover (Beweisführer) und Verifier (Verifizierer). Der Prover versucht mittels eines Beweissystems dem Verifier die Korrektheit seiner Aussage zu beweisen.

Wenn die Kommunikation zwischen Prover und Verifier interaktiv abläuft, so nennt man dies ein *interaktives Beweissystem*, welches nach dem Challenge-Response-Prinzip aufgebaut ist. Der Prover schickt dabei eine Response auf die vom Verifier geschickte Challenge.

Ein Beweissystem muss durchführbar und korrekt sein. *Durchführbarkeit* bedeutet: Der Verifier akzeptiert den Beweis des Provers immer, wenn der Prover ehrlich ist und beide das Protokoll korrekt befolgen. Ein interaktiver Beweis ist *korrekt*, wenn ein unehrlicher Prover, der versucht eine falsche Behauptung zu beweisen, mit sehr hoher Wahrscheinlichkeit scheitert.

Zero-Knowledge-Beweis

In einem Zero-Knowledge-Beweis[3] erfährt der Verifier nur, dass der Prover den Beweis einer Behauptung kennt. Der Prover gibt in keiner Weise sein Wissen preis.

Für einen Zero-Knowledge-Beweis müssen die Eigenschaften *Durchführbarkeit* und *Korrektheit* des interaktiven Beweissystems gelten. Zusätzlich muss noch die *Zero-Knowledge-Eigenschaft* gelten: Aus der Interaktion zwischen dem Prover und dem Verifier darf nicht mehr Wissen als die (Un-)Gültigkeit der zu beweisenden Aussage gewonnen werden. Ein Dritter, der die Interaktion zwischen Prover und Verifier verfolgt, erfährt nicht einmal, ob der Prover überhaupt das Geheimnis kennt (oder die Interaktion zwischen Prover und Verifier abgesprochen war).

3.4.1. Proof of Knowledge eines diskreten Logarithmus

Das folgende Protokoll nach Schnorr [6] erlaubt den Proof of Knowledge des Zufallswertes k der Verschlüsselung. Gegeben sei eine ElGamal-Verschlüsselung $(x, y) = (g^k, h^k m)$.

- Der Prover legt sich auf die Zufälligkeit $w \in_R \mathbb{Z}_q$ fest und schickt $A = g^w \bmod p$ (Commitment) an den Verifier.
- Der Verifier antwortet mit einer zufällig gewählten Challenge $c \in_R \mathbb{Z}_q$.
- Nach Erhalt von c schickt der Prover $r = w + ck \pmod q$ (Response).

Der Verifier akzeptiert den Beweis nur, wenn $g^r = g^{w+ck} = g^w \cdot g^{ck} = A(g^k)^c = Ax^c$.

Die Reihenfolge der ausgetauschten Messages ist relevant. Das Tripel (A, c, r) heisst eine *accepting conversation*.

Protokolle mit obiger 3-Wege-Struktur (Commitment, Challenge, Response) werden *Sigma-Protokolle* gennant. Das griechische Zeichen Σ visualisiert dabei den Protokoll-Fluss.

3.4.2. Beweis der Gleichheit von diskreten Logarithmen

Dieses Protokoll nach Chaum und Pedersen [7] ist ein Beweis für die Relation

$$\log_g(x) = \log_h(y) \tag{3.1}$$

für zwei Werte $x = g^\alpha$ und $y = h^\alpha$. Der Prover demonstriert damit sein Wissen von α und der Beweis stellt sicher, dass x und y den gleichen diskreten Logarithmus α zur Basis g und h haben.

[3] engl. Zero-Knowledge-Proof

1. Der Prover legt sich auf die Zufälligkeit $w \in \mathbb{Z}_q$ fest (Commitment) und schickt $a = g^w \bmod p$ und $b = h^w \bmod p$ an den Verifier.

2. Der Verifier antwortet mit einer zufällig gewählten Challenge $c \in \mathbb{Z}_q$.

3. Nach Erhalt von c schickt der Prover $r = w + c\alpha \bmod p$ (Response).

Der Verifier V akzeptiert den Proof nur, wenn $g^r = ax^c \bmod p$ und $h^r = by^c \bmod p$.

3.4.3. Nicht-interaktiver Zero-Knowledge-Beweis

Das Problem beim Zero-Knowledge-Beweis ist, dass interaktive Beweise nur den einzigen beteiligten Verifier überzeugen. Allerdings sollte die Korrektheit des Protokolls öffentlich überprüfbar sein. Deshalb werden die Beweise nicht-interaktiv gemacht, indem man die Fiat-Shamir-Heuristik anwendet. Dabei erstellt der Prover die Challenges selbst mit Hilfe einer Hash-Funktion, welche das zufällige Auswählen simuliert. Aufgrund der Einweg-Eigenschaft der Hashfunktion kennt der Prover die Challenge erst, wenn er die Commitments gebildet hat und in die Hashfunktion eingegeben hat.

1. Beim Protokoll von Schnorr [6] würde der Prover die Challenge selber erstellen: $c = hash(A, x, g) \bmod q$. Der nicht-interaktive Proof of Knowledge wäre dann (A, c, r).

2. Beim Protokoll von Chaum und Pedersen würde die Challenge mit $c = hash(h, y, a, b) \bmod q$ berechnet werden. Der nicht-interaktive Proof of Knowledge wäre dann (a, b, c, r).

3.4.4. Gültigkeitsbeweis

Ein weiterer Proof of Knowledge soll sicherstellen, dass der Prover einen Plaintext m_i aus einer Menge $M = (m_1, m_2, \ldots, m_L)$ von L erlaubten Plaintexten mit ElGamal verschlüsselt hat, ohne verraten zu müssen, welches m_i er verschlüsselt hat, wobei der Ciphertext $(x, y) = (g^k, h^k m_i)$ ist. Dies kann mit Hilfe einer OR-Komposition von L Proofs of Knowledge gemäss [8] sichergestellt werden. Dies beweist die Relation

$$log_g x = log_h(y/m_1) \vee \ldots \vee log_g x = log_h(y/m_L) \tag{3.2}$$

1. Der Prover wählt eine zufällige Zahl $w \in_R \mathbb{Z}_q$ und berechnet $a_i = g^w \bmod p$ und $b_i = h^w \bmod p$

2. Für jedes $j = 1, \ldots, i-1, i+1, \ldots, L$ wählt der Prover ein zufälliges $r_j, d_j \in_R \mathbb{Z}_q$ und berechnet $a_j = g^{r_j} x^{d_j} \bmod p$ und $b_j = h^{r_j}(y/m_j)^{d_j} \bmod p$

3. Der Prover berechnet $c = H(a_1, b_1, \ldots, a_L, b_L) \bmod q$, mit der Hashfunktion $H()$
4. Der Prover berechnet $d_i = c - \sum_{j \neq i} d_j \bmod q$ und $r_i = w - kd_i \bmod q$
5. Der Prover veröffentlicht $(A, B, D, R) = (a_1, b_1, d_1, r_1, ..., a_L, b_L, d_L, r_L)$

Der Verifier überprüft nun, ob $d_1 + \ldots + d_L = H(g^{r_1} h^{d_1}, {h_1}^r (y/m_1)^{d_1}, \ldots, g^{r_L} h^{d_L}, {h_L}^r (y/m_L)^{d_L}$

4. Projektspezifische Abläufe

Dieses E-Voting-System beruht darauf, dass verschiedene Person diverse Rollen einnehmen können. Es handelt sich dabei um einen Administrator, mindestens zwei Trustees und mindestens einen Wähler. Dem Administrator ist es erlaubt, eine Abstimmung einzurichten und er definiert auch die beteiligten Trustees und bestimmt, welche Wähler an der Abstimmung teilnehmen dürfen. Die Trustees sind für die Generierung des Schlüsselmaterials sowie die Entschlüsselung des Resultats zuständig. Die Wähler schlussendlich sind diejenigen Personen, dene es erlaubt ist, an der Abstimmung teilzunehmen.

Zwischen Erstellung einer Abstimmung bis zum Auszählen der Stimmen und Veröffentlichung des Resultats geschehen viele Schritte, die in diesem Kapitel detailliert beschrieben werden. Die folgende Liste zeigt die Abläufe in einer groben Übersicht.

- Der Administrator erfasst eine Option und dazugehörende Optionen
- Der Administrator definiert die beteiligten Trustees und einen Threshold
- Jeder Trustee berechnet und veröffentlicht seine Parameter
- Jeder Trustee überprüft die Parameter der anderen Trustees
- Der Administrator gibt die Abstimmung für die Stimmabgabe frei
- Die Wähler geben ihre Stimme ab
- Der Administrator schliesst die Abstimmung und summiert die Stimmen
- Die Trustees führen eine Teilentschlüsselung des Abstimmungsresultats durch
- Sobald genug Teilentschlüsselungen durchgeführt wurden, kann das Resultat entschlüsselt und veröffentlicht werden

Das folgende Aktivitätsdiagramm visualisiert die beschriebenen Aktionen im System in chronologischer Reihenfolge. Die hier referenzierten Use Cases (UC) sind im Anhang dieses Dokuments beschrieben.

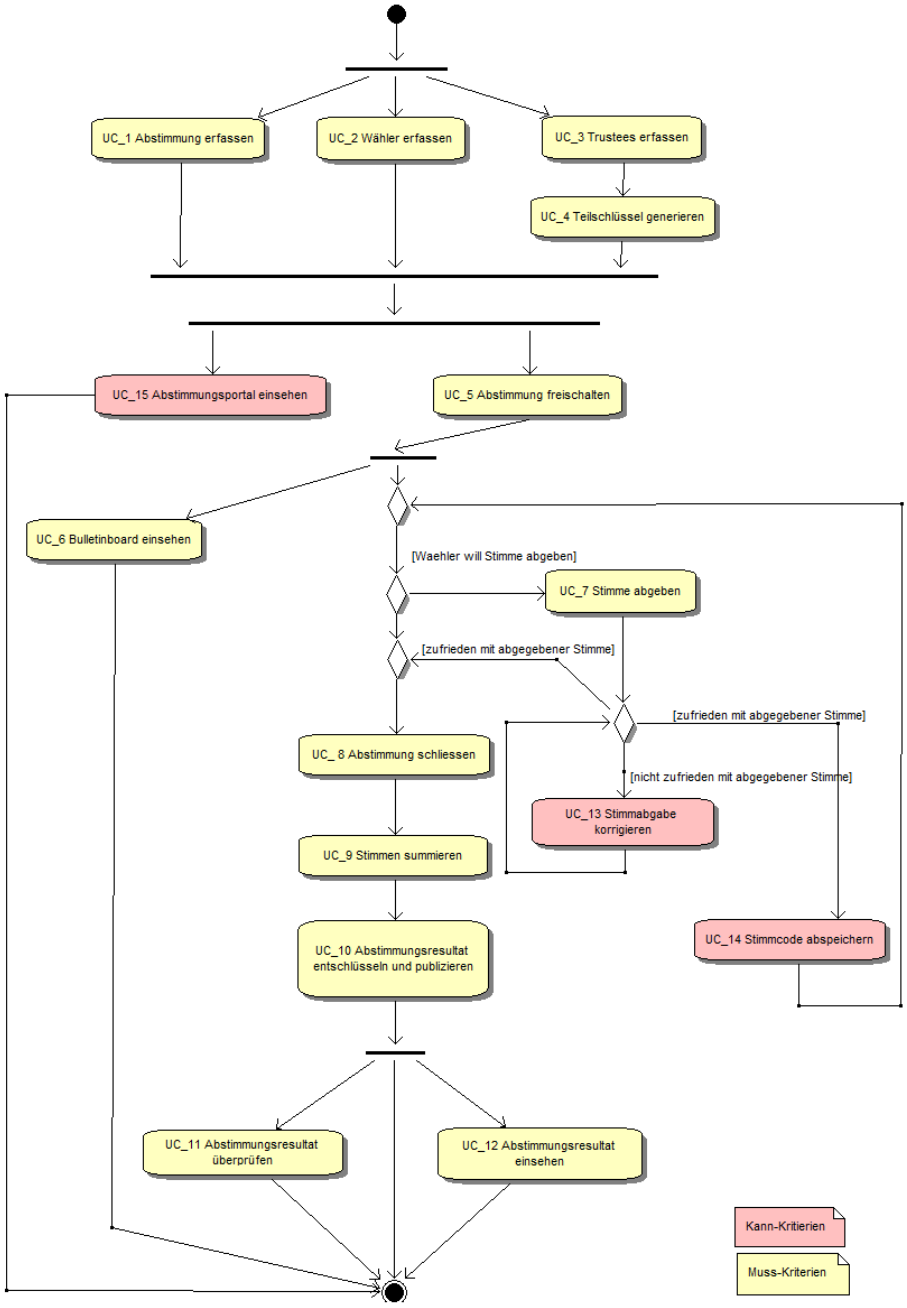

Abbildung 4.1.: Aktivitätsdiagramm

Zur Veranschaulichung wird eine Abstimmung durchgespielt und dabei die Berechnungen und Daten aufgeführt, die im Lebenszyklus dieser Abstimmung entstehen. Das Szenario besteht darin, dass ein Administrator eine Abstimmung einrichtet und die Auswahloptionen definiert. Der Administrator benennt zudem die Trustees und definiert einen Threshold. Anschliessend geben die Wähler je eine Stimme ab. Diese Stimmen

werden ausgezählt und das Resultat veröffentlicht. Für die ElGamal-Verschlüsselung werden die Domainparameter $p = 383$, $q = 191$ und $g = 2$ definiert. Da es sich um ein anschauliches Beispiel handeln soll und für dessen Übersichtlichkeit, wurden die Domänenparameter bewusst so klein gewählt. In der Praxis sollten sehr viel grössere Werte verwendet werden.

4.1. Erstellen einer Abstimmung

Der Administrator muss zuerst eine Abstimmung erstellen, welche die Fragestellung und die möglichen Wahloptionen beinhaltet. Er definiert auch den Threshold t, im Beispiel wurde $t = 3$ gewählt. Zu diesem Zeitpunkt ist die Abstimmung noch nicht für die Stimmabgabe geöffnet und auch der für die Verschlüsselung benötigte öffentliche Schlüssel (Public Key) ist noch nicht definiert.

Der Administrator erstellt die zur Auswahl stehenden Optionen. Für jede Option wird später das Resultat in verschlüsselter Form berechnet, bis zuletzt das Schlussresultat entschlüsselt und veröffentlicht wird.

4.2. Erstellen des Schlüsselmaterials

Die Abstimmung ist immer noch nicht für die Stimmabgabe freigegeben, da zuerst das Schlüsselmaterial erstellt werden muss.

Dazu wählt jeder Trustee A_j zufällig Koeffizienten a_i aus $\mathbb{Z}_p^*$ für seine Funktion f_j vom Grad $t-1$:

$$f_j = a_{j0} + a_{j1}x + a_{j2}x^2 + \ldots + a_{jt-1}x^{t-1}$$

Trustee A_j	Funktion f_j	a_{j0}	a_{j1}	a_{j2}
Trustee 1	$f_1 = 189 + 136x + 154x^2$	$a_{10} = 189$	$a_{11} = 136$	$a_{12} = 154$
Trustee 2	$f_2 = 119 + 147x + 119x^2$	$a_{20} = 119$	$a_{21} = 147$	$a_{22} = 119$
Trustee 3	$f_3 = 179 + 189x + 175x^2$	$a_{30} = 179$	$a_{31} = 189$	$a_{32} = 175$
Trustee 4	$f_4 = 96 + 121x + 139x^2$	$a_{40} = 96$	$a_{41} = 121$	$a_{42} = 139$
Trustee 5	$f_5 = 154 + 50x + 66x^2$	$a_{50} = 154$	$a_{51} = 50$	$a_{52} = 66$

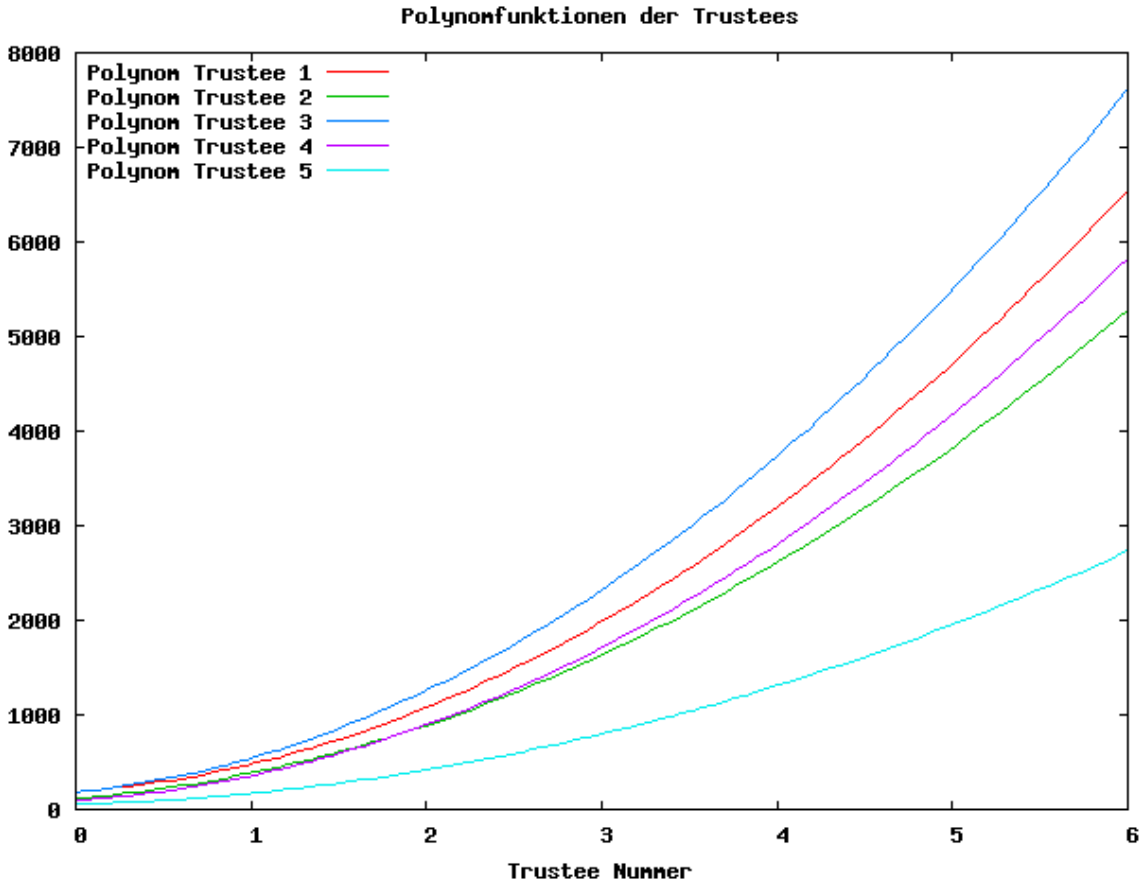

Abbildung 4.2.: Graphen von f_j

Jeder Trustee A_j veröffentlicht die Werte $g^{a_i} \bmod p$, wobei a_{ji} die Koeffizienten der Funktion f_j sind.

Trustee A_j	$g^{a_{j0}} \bmod p$	$g^{a_{j1}} \bmod p$	$g^{a_{j2}} \bmod p$
Trustee 1	$2^{189} \bmod 383 = 96$	$2^{136} \bmod 383 = 100$	$2^{154} \bmod 383 = 348$
Trustee 2	$2^{119} \bmod 383 = 304$	$2^{147} \bmod 383 = 278$	$2^{119} \bmod 383 = 304$
Trustee 3	$2^{179} \bmod 383 = 36$	$2^{189} \bmod 383 = 96$	$2^{175} \bmod 383 = 98$
Trustee 4	$2^{96} \bmod 383 = 137$	$2^{121} \bmod 383 = 67$	$2^{139} \bmod 383 = 34$
Trustee 5	$2^{154} \bmod 383 = 348$	$2^{50} \bmod 383 = 138$	$2^{66} \bmod 383 = 189$

Jeder Trustee berechnet mit seiner Funktion $f_j(i) \bmod q$ Funktionswerte (sogenannte "Shares") für jeden anderen Trustee A_i, wobei $f_j(0) = a_{j0}$:

Trustee A_j	$f_j(0) \bmod q$	$f_j(1) \bmod q$	$f_j(2) \bmod q$
Trustee 1	$f_1(0) \bmod 191 = 189$	$f_1(1) \bmod 191 = 97$	$f_1(2) \bmod 191 = 122$
Trustee 2	$f_2(0) \bmod 191 = 119$	$f_2(1) \bmod 191 = 3$	$f_2(2) \bmod 191 = 125$
Trustee 3	$f_3(0) \bmod 191 = 179$	$f_3(1) \bmod 191 = 161$	$f_3(2) \bmod 191 = 111$
Trustee 4	$f_4(0) \bmod 191 = 96$	$f_4(1) \bmod 191 = 165$	$f_4(2) \bmod 191 = 130$
Trustee 5	$f_5(0) \bmod 191 = 154$	$f_5(1) \bmod 191 = 79$	$f_5(2) \bmod 191 = 136$

Trustee A_j	$f_j(3) \bmod q$	$f_j(4) \bmod q$	$f_j(5) \bmod q$
Trustee 1	$f_1(3) \bmod 191 = 73$	$f_1(4) \bmod 191 = 141$	$f_1(5) \bmod 191 = 135$
Trustee 2	$f_2(3) \bmod 191 = 103$	$f_2(4) \bmod 191 = 128$	$f_2(5) \bmod 191 = 9$
Trustee 3	$f_3(3) \bmod 191 = 29$	$f_3(4) \bmod 191 = 106$	$f_3(5) \bmod 191 = 151$
Trustee 4	$f_4(3) \bmod 191 = 182$	$f_4(4) \bmod 191 = 130$	$f_4(5) \bmod 191 = 165$
Trustee 5	$f_5(3) \bmod 191 = 134$	$f_5(4) \bmod 191 = 73$	$f_5(5) \bmod 191 = 144$

Da diese Werte geheim bleiben müssen, resp. nur vom Empfänger gelesen werden dürfen, wird mit Public Key Kryptographie gearbeitet und die Werte, die jeder A_j jedem anderen Trustee A_i sendet, werden mit dem Public Key des Trustees A_i verschlüsselt.

Sobald nun mindestens t Trustees untereinander die "Shares" ausgetauscht haben, überprüft jeder Trustee die erhaltenen Messages, indem jeder Trustee A_j dieser t Trustees überprüft, ob die von Trustee A_i erhaltenen "Shares" mit den von Trustee A_i publizierten Koeffizienten übereinstimmen. Schlägt die Überprüfung fehl, wird ein Complaint gegen den fehlbaren Trustee publiziert. Wenn mehr als $t-1$ Trustees gegen den fehlbaren Trustee einen Complaint veröffentlicht haben, wird dieser disqualifiziert und vom Abstimmungsprozess ausgeschlossen.

Jeder A_i überprüft also die von den anderen Trustees A_j erhaltenenen Werte $f_j(i)$ mit

$$g^{f_j(i)} \bmod p = \prod_{l=0}^{t-1} g^{a_{jl} \cdot l \cdot i} \bmod p$$

wobei $g^{a_{jl}}$ bereits vorhanden sind. In der folgenden Tabelle sieht man, wie der Trustee 1 die Werte der anderen Trustees überprüft. Die restlichen Trustees machen die Prüfung analog.

Trustee A_j	$g^{a_{jl}} \bmod p$	$\prod$
Trustee 1	$2^{97} \bmod 383 = 274$	$96^{1^0} \cdot 100^{1^1} \cdot 348^{1^2} \bmod 383 = 274$
Trustee 2	$2^{161} \bmod 383 = 116$	$36^{1^0} \cdot 96^{1^1} \cdot 98^{1^2} \cdot \bmod 383 = 116$
Trustee 3	$2^{79} \bmod 383 = 202$	$348^{1^0} \cdot 138^{1^1} \cdot 189^{1^2} \cdot \bmod 383 = 202$
Trustee 4	$2^{165} \bmod 383 = 324$	$137^{1^0} \cdot 67^{1^1} \cdot 34^{1^2} \cdot \bmod 383 = 324$
Trustee 5	$2^3 \bmod 383 = 8$	$304^{1^0} \cdot 278^{1^1} \cdot 304^{1^2} \cdot \bmod 383 = 8$

Zudem berechnet jeder Trustee seinen privaten Teilschlüssel (Private Key-Part) indem

er alle erhaltenen “Shares” addiert:

$$s_j = \sum f_i(j) \bmod q$$

Trustee A_j	s_j
Trustee 1	$97 + 3 + 161 + 165 + 79 \bmod 191 = 123$
Trustee 2	$122 + 125 + 111 + 130 + 136 \bmod 191 = 51$
Trustee 3	$73 + 103 + 29 + 182 + 134 \bmod 191 = 139$
Trustee 4	$141 + 128 + 106 + 130 + 73 \bmod 191 = 5$
Trustee 5	$135 + 9 + 151 + 165 + 144 \bmod 191 = 31$

Jeder Trustee committet sich zu seinem privaten Teilschlüssel, indem er den dazugehörenden öffentlichen Teilschlüssel (Public Key-Part) $h_j = g^{a_j} \bmod p$ veröffentlicht.

Trustee A_j	h_j
Trustee 1	$2^{123} \bmod 383 = 268$
Trustee 2	$2^{51} \bmod 383 = 276$
Trustee 3	$2^{139} \bmod 383 = 34$
Trustee 4	$2^{5} \bmod 383 = 32$
Trustee 5	$2^{31} \bmod 383 = 350$

Sobald mindestens t gültige “Shares” vorhanden sind, kann der Administrator die Abstimmung eröffnen. Dabei wird der öffentliche Schlüssel h der Abstimmung mit Hilfe der veröffentlichten Koeffizienten jedes Trustees berechnet:

$$h = \prod_{j=1}^{n} g^{a_{j0}} \bmod p = 96 \cdot 304 \cdot 36 \cdot 137 \cdot 348 \bmod 383 = 162$$

4.3. Stimmabgabe

Die Abstimmung ist nun geöffnet und der öffentliche Schlüssel für die Abstimmung wurde erstellt. Somit können die Wähler zur Stimmabgabe schreiten. Dabei wird pro Wahloption entweder der Wert g^0 (Nein) oder g^1 (Ja) mit dem öffentlichen Schlüssel des ElGamal-Kryptosystems verschlüsselt. Der Grund für die Exponentiation zur Basis g besteht darin, dass ElGamal eine multiplikativ homomorphe Eigenschaft besitzt. Da wir aber bei der Auszählung die Summe der Stimmen und nicht deren Produkt benötigen, ist diese Operation notwendig. Zusätzlich wird für jede verschlüsselte Option ein Zero-Knowledge-Beweis erstellt, der beweist, dass entweder 0 oder 1 unter Verwendung des verfügbaren öffentlichen Schlüssel verschlüsselt worden ist.

In folgender Tabelle stehen die ElGamal Ciphertexte (x, y) und die Werte des Zero-Knowledge-Beweises für jede der drei zur Auswahl stehenden Optionen einer Stimmabgabe, bei welcher der Wähler sich für die Option 1 entschieden hat. Für jede Option wird dabei ein zufälliges $k \in_R \mathbb{Z}_q$ gewählt und damit die ElGamal-Verschlüsselung durchgeführt.

Option	k	x	y
1	186	$2^{186} \bmod 383 = 12$	$162^{186} \cdot 2^1 \bmod 383 = 100$
2	112	$2^{112} \bmod 383 = 146$	$162^{112} \cdot 2^0 \bmod 383 = 317$
3	155	$2^{155} \bmod 383 = 313$	$162^{155} \cdot 2^0 \bmod 383 = 86$

Für die gewählte Option 1, also einer "Ja"-Stimme, folgen nun die Berechnungen eines Proof of Validity. Dabei berechnet der Wähler zuerst a_2 und b_2 mit einem zufällig gewähltem $w \in_R \mathbb{Z}_q$:

$$a_2 = g^w \bmod p = 2^{163} \bmod 383 = 81$$

$$b_2 = 162^{163} \bmod 383 = 193$$

Zudem wählt er $r_1 = 51$ und $d_1 = 51$ zufällig aus $\mathbb{Z}_q$ und berechnet damit a_1 und b_1 für die Stimme m_j, welche bei einer "Ja"-Stimme g^1 und bei einer "Nein"-Stimme g^0 ist.

$$a_1 = g^{r_1} \cdot x^{d_1} \bmod p = 2^{51} \cdot 12^{51} \bmod 383 = 18$$

$$b_1 = h^{r_1}(y/m_j)^{d_1} \bmod p = 162^{51}(100/(2^0))^{51} \bmod 383 = 305$$

Weiter wird der Wert c benötigt, um den Proof nicht-interaktiv zu machen. Dies geschieht mit einer Hashfunktion $H()$, in unserem E-Voting-System wird SHA-1 eingesetzt:

$$c = H(a_1, b_1, a_2, b_2) \bmod q = H(18, 305, 81, 193)^1 \bmod 191 = 79$$

Mit diesen Werten wird schlussendlich d_2 und r_2 berechnet.

$$d_2 = c - d_1 \bmod q = 79 - 51 \bmod 191 = 28$$

$$r_2 = w - k \cdot d_2 \bmod q = 163 - 186 * 28 \bmod 191 = 112$$

[1] SHA-1(18, 305, 81, 193) = -712011095468043117372801719815508208220099304112

Der Wähler veröffentlicht nun die Werte $a_1, a_2, b_1, b_2, d_1, d_2, r_1$ und r_2 für alle Optionen:

Option	a1	a2	b1	b2	d1	d2	r1	r2
1	18	81	305	193	51	28	51	112
2	75	343	129	43	118	4	90	4
3	169	113	153	292	132	146	29	146

Um sicherzustellen dass der Wähler genau eine Option gewählt hat, also dass die Summe aller Optionen 1 ist, wird ein weiterer Zero-Knowledge-Beweis über alle Optionen erstellt. Hierbei wird die $k = \sum k_i$, $x = \prod x_i$ und $y = \prod y_i$ von allen Optionen gebildet.

Dabei berechnet der Wähler zuerst a_2 und b_2 mit einem zufällig gewähltem $w \in_R \mathbb{Z}_q$:

$$a_2 = g^w \bmod p = 2^{133} \bmod 383 = 204$$

$$b_2 = 162^{133} \bmod 383 = 372$$

Zudem wählt er $r_1 = 175$ und $d_1 = 175$ zufällig aus $\mathbb{Z}_q$ und berechnet damit a_1 und b_1

$$a_1 = g^{r_1} \cdot (\prod x_i)^{d_1} \bmod p = 2^{175} \cdot (12 \cdot 146 \cdot 313)^{175} \bmod 383 = 6$$

$$b_1 = h^{r_1} (\prod y_i / m_j)^{d_j} \bmod p = 162^{175} ((100 \cdot 317 \cdot 86)/(2^0))^{175} \bmod 383 = 139$$

Weiter wird der Wert c benötigt, um den Proof nicht-interaktiv zu machen. Dies geschieht mit einer Hashfunktion $H()$, in unserem E-Voting-System wird SHA-1 eingesetzt:

$$c = H(a_1, b_1, a_2, b_2) \bmod q = H(6, 139, 204, 372)^2 \bmod 191 = 4$$

Mit diesen Werten wird schlussendlich d_2 und r_2 berechnet.

$$d_2 = c - d_1 \bmod q = 4 - 175 \bmod 191 = 20$$

$$r_2 = w - \sum k \cdot d_2 \bmod q = 175 - (186 + 112 + 115) \cdot 28 \bmod 191 = 50$$

Der Wähler veröffentlicht nun die Werte $a_1, a_2, b_1, b_2, d_1, d_2, r_1$ und r_2 des Proofs.

[2]SHA-1(6, 139, 204, 372) = 90881597797110233107600776315107933871509714432

a1	a2	b1	b2	d1	d2	r1	r2
372	6	301	232	174	78	174	134

Mittels SHA-1 wird der Hashwert der Stimme berechnet damit der Wähler sicher sein kann, dass nach der Stimmabgabe die Werte seiner Stimme nicht verändert worden sind.

4.4. Auszählung und Veröffentlichung des Resultats

Nach der Stimmabgabe schliesst der Administrator die Abstimmung und löst das "Tallying", also die homomorphe Summierung aller Stimmen aus. Dabei werden die Zero-Knowledge-Beweise der Stimmen überprüft und alle gültigen Stimmen in die Auszählung einbezogen.

Dazu wird der Proof jeder Option überprüft. Als Beispiel folgt die Überprüfung der Option 1:

$$p_1 = g^{r_1} \cdot x^{d_1} \bmod p = 2^{51} \cdot 12^{51} \bmod 383 = 18$$

$$p_2 = h^{r_1} \cdot (y/g^0)^{d_1} \bmod p = 162^{51} \cdot (100/2^0)^{51} \bmod 383 = 305$$

$$p_3 = g^{r_2} \cdot x^{d_2} \bmod p = 2^{112} \cdot 12^{28} \bmod 383 = 81$$

$$p_4 = h^{r_2} \cdot (y/g^1)^{d_2} \bmod p = 162^{112} \cdot (100/2^1)^{28} \bmod 383 = 193$$

$$H(18, 305, 81, 193)^3 \bmod q \stackrel{?}{=} d_1 + d_2 \bmod q = 51 + 28 \bmod 191 = 79$$

Nun muss noch der Proof überprüft werden, der über alle Optionen gebildet wurde. Hierzu wird $x = \prod x_i$ und $y = \prod y_i$ von allen Optionen gebildet.

$$p_1 = g^{r_1} \cdot (\prod x_i)^{d_1} \bmod p = 2^{175} \cdot (12 \cdot 146 \cdot 313)^{175} \bmod 383 = 6$$

$$p_2 = h^{r_1} \cdot (\prod y_i/g^0)^{d_1} \bmod p = 162^{175} \cdot ((100 \cdot 317 \cdot 86)/2^0)^{175} \bmod 383 = 139$$

[3] $SHA1(18, 305, 81, 193) = -712011095468043117372801719815508208220099304112$

$$p_3 = g^{r_2} \cdot (\prod x_i)^{d_2} \bmod p = 2^{50} \cdot (12 \cdot 146 \cdot 313)^{20} \bmod 383 = 204$$

$$p_4 = h^{r_2} \cdot (\prod y_i / g^1)^{d_2} \bmod p = 162^{50} \cdot ((100 \cdot 317 \cdot 86)/2^1)^{20} \bmod 383 = 372$$

$$H(6, 139, 204, 372)^4 \bmod q \stackrel{?}{=} d_1 + d_2 \bmod q = 175 + 20 \bmod 191 = 4$$

Das Resultat entsteht durch homomorphes Summieren der Stimmen pro Option. Dabei erhält man pro Option einen ElGamal-Ciphertext mit den Werten x und y, der die Summe der Ja-Stimmen in verschlüsselter Form beinhaltet.

Im folgenden Beispiel haben zwei Wähler für die Option 1 gestimmt und eine Person für die Option 2.

Wähler	Option	x	y
1	1	12	100
1	2	146	317
1	3	313	86
2	1	119	372
2	2	17	130
2	3	169	153
3	1	42	55
3	2	330	4
3	3	284	38

Die Produkte der x und y Werte pro Option ergeben folgende Werte.

Option	x	y
1	$12 \cdot 119 \cdot 42 \bmod 383 = 228$	$100 \cdot 372 \cdot 55 \bmod 383 = 14$
2	$146 \cdot 17 \cdot 330 \bmod 383 = 206$	$317 \cdot 130 \cdot 4 \bmod 383 = 150$
3	$313 \cdot 169 \cdot 284 \bmod 383 = 339$	$86 \cdot 153 \cdot 38 \bmod 383 = 189$

Nun müssen die Trustees eine "Distributed Decryption" des verschlüsselten Resultats durchführen. Jeder Trustee führt dazu eine Teil-Entschlüsselung jeder verschlüsselten und summierten Option aus, indem er $w = x^{s_j}$ rechnet. Für jedes w pro Option erstellt er einen Zero-Knowledge-Beweis mit dessen Hilfe überprüft werden kann, dass der Ciphertext mit seinem privaten Teilschlüssel des Trustees berechnet wurde. Der Trustee veröffentlicht w und den Zero-Knowledge-Beweis mit den Werten a, b, c und r.

[4] $SHA1(6, 139, 204, 372) = 90881597797110233107600776315107933871509714432$

Sobald pro Option mindestens t gültige Teil-Entschlüsselungen vorhanden sind, wird mittels Lagrange-Interpolation der Ciphertext entschlüsselt, indem man

$$g^m = y / \prod_{n=1}^{t} {w_{jn}}^{\lambda_j} \bmod p$$

berechnet.

In unserem Beispiel berechnen nun drei der fünf Trustees pro Option die folgenden Teil-Resultate.

Trustee	Option 1	Option 2	Option 3	λ
3	$228^{139} \bmod 383 = 258$	$206^{139} \bmod 383 = 48$	$339^{139} \bmod 383 = 227$	10
4	$228^{5} \bmod 383 = 294$	$206^{5} \bmod 383 = 370$	$339^{5} \bmod 383 = 129$	176
5	$228^{31} \bmod 383 = 81$	$206^{31} \bmod 383 = 268$	$339^{31} \bmod 383 = 130$	6

Betrachten wir die Entschlüsselung der Option 1.

$$g^m = y / \prod_{n=1}^{t} {w_{jn}}^{\lambda_j} \bmod p = 14/(258^{10} \cdot 294^{176} \cdot 81^{6}) \bmod 383 = 4$$

Mittels "Brute Force" ermitteln wir m aus g^m, also die Summe der Stimmen, weil kein effizienter Algorithmus bekannt, um den diskreten Logarithmus zu finden. Sind n Stimmen eingegangen, ist $m \leq n$ und die Komplexität für das Finden von m in $O(n)$ lösbar. In unserem Beispiel erhalten wir als Resultat zwei Stimmen für die Option 1. Analog dazu wird die Anzahl Stimmen für die restlichen Optionen berechnet.

Somit haben wir das Abstimmungsresultat erfolgreich entschlüsselt.

5. Implementation

Die Aufgabe dieser Bachelor-Thesis besteht darin, einen Prototypen eines E-Voting Systems gemäss dem CGS97 Protokoll [1] zu implementieren. Entstanden ist dabei ein Produkt, das "Comitia" getauft wurde. Das lateinische Wort "Comitia" wurde im antiken Rom als Bezeichnung für eine Volksversammlung verwendet, an der auch Entscheidungen gefällt wurden.

Das Ziel von "Comitia" ist es, die komplexen kryptographischen Abläufe des CGS97 Protokolls [1] in eine anwenderfreundliche Applikation zu verpacken, so dass der Vorgang der Stimmabgabe für einen Wähler möglichst einfach ist. Dieser Faktor ist nicht zu vernachlässigen, wenn es darum geht, die Akzeptanz von E-Voting-Systemen in der Bevölkerung steigern zu wollen.

Damit der Wähler für eine Stimmabgabe keine zusätzliche Software auf seinem Computer installieren muss, wurde diese Funktionalität in Form einer Web-Applikation programmiert, welche im Browser des Wählers ausführbar ist.

In diesem Kapitel wird der Aufbau dieses Softwareprodukts im Detail erklärt.

5.1. Eingesetzte Software und Programmierumgebung

Programmiert wurde "Comitia" ausschliesslich in Java. Diese weit verbreitete und vielseitige Programmiersprache bietet viele vorgefertigte Bausteine und Konzepte, die sehr einfach in einem eigenen Softwareprojekt verwendet werden können. Ausserdem bieten viele Entwickler der grossen Community rund um Java frei verfügbare Erweiterungen an, die ebenfalls eingebunden werden können. Zudem ist Java für alle gängigen Betriebssystemen verfügbar und kostenlos erhältlich.

5.1.1. Apache Tomcat

Apache Tomcat ist ein Servletcontainer, worin sich Java Code als Webapplikation ausführen lässt. Er fungiert als Webserver und bietet somit die Schnittstelle, über die ein Benutzer mit "Comitia" interagieren kann. Als Transport-Protokoll wird HTTP verwendet, wovon es auch eine sichere Variante gibt, bei welcher jeglicher Datenverkehr mittels SSL/TLS

verschlüsselt wird (HTTPS). Tomcat kümmert sich ebenfalls um die Verschlüsselung dieses Datenverkehrs. Apache Tomcat ist Open Source Software und somit kostenlos verfügbar.

5.1.2. MySQL

MySQL ist der Name eines frei verfügbaren relationalen Datenbanksystems der Firma Oracle. Die grossen Vorteile liegen nebst der kostenlosen Verfügbarkeit in der hohen Geschwindigkeit, der Einfachheit, der Zuverlässigkeit und nicht zuletzt der grossen Verbreitung und dem damit verbundenem Support der Anwender-Community. Ausserdem arbeiten Java und MySQL problemlos zusammen, was für dieses Projekt unabdingbar ist.

5.1.3. Apache Cayenne

Apache Cayenne ist ein Open Source Projekt, welches ein ORM Framework implementiert. Mit dessen Hilfe lassen sich Java Objekte in eine relationale Datenbankstruktur abbilden und umgekehrt. Das bietet den Vorteil, dass nur noch mit Java Objekten gearbeitet wird und es entfällt die fehleranfällige manuelle Erstellung von SQL Befehlen. Cayenne kann mit vielen Datenbanksystemen zusammenarbeiten, unter anderem auch mit MySQL das in “Comitia” zum Einsatz kommt.

5.1.4. Google Web Toolkit

Das Google Web Toolkit (GWT) ist ein von der Firma Google veröffentlichtes Framework zur Erstellung von interaktiven Web 2.0 Applikationen. Die Besonderheit dieses Frameworks ist ein Compiler, der Java Code in JavaScript umwandelt, welches von den Browsern ausgeführt werden kann. Der entscheidende Vorteil dabei ist, dass das gesamte Projekt mit Java implementiert werden kann und keine grundlegende Unterscheidung zwischen client- und serverseitigem Code gemacht werden muss.

5.1.5. SCM und ANT

Das Source-Code-Management wurde durch Subversion unterstützt. Das zentrale Repository liegt auf der Kollaborations-Plattform “Origo”, welche von der ETH Zürich zur Verfügung gestellt wird. Der Build-Vorgang wird mit Hilfe von ANT automatisiert.

5.1.6. Eclipse IDE

Für die Implementation dieses Projekts wurde die Entwicklungsumgebung Eclipse eingesetzt. Eclipse ist ein Open Source Projekt und auch kostenlos erhältlich. Eclipse ist eines der bekanntesten Entwicklungs-Werkzeugen, wenn es um die Programmierung von Java geht. Google hat ausserdem zusammen mit GWT ein Plugin für Eclipse veröffentlicht, das die Entwicklung von GWT Applikationen in Eclipse sehr vereinfacht.

5.2. Architektur

Grundsätzlich wurde die Software als klassische Client-Server Applikation implementiert. Somit besteht "Comitia" serverseitig aus einer zentralen Applikationsinstanz, welche die Daten in der ihr angebundenen Datenbank verwaltet und für die Clients aufbereitet. Ausserdem werden Schnittstellen zur kontrollierten Manipulation dieser Daten exponiert. Als Client wird ein handelsüblicher Browser benutzt, welcher den Benutzern eine webbasierte Bedienungsoberfläche bietet. Eine Ausnahme musste beim Benutzerinterface für den Trustee gemacht werden. Da die Rechenoperationen, die ein Trustee ausführen muss, ziemlich rechenintensiv sind und die Leistungsfähigkeit von clientseitigem Code im Browser (JavaScript) zum heutigen Zeitpunkt nicht sehr performant ist, wurde ein zusätzliches Java Programm mit graphischem Benutzerinterface erstellt.

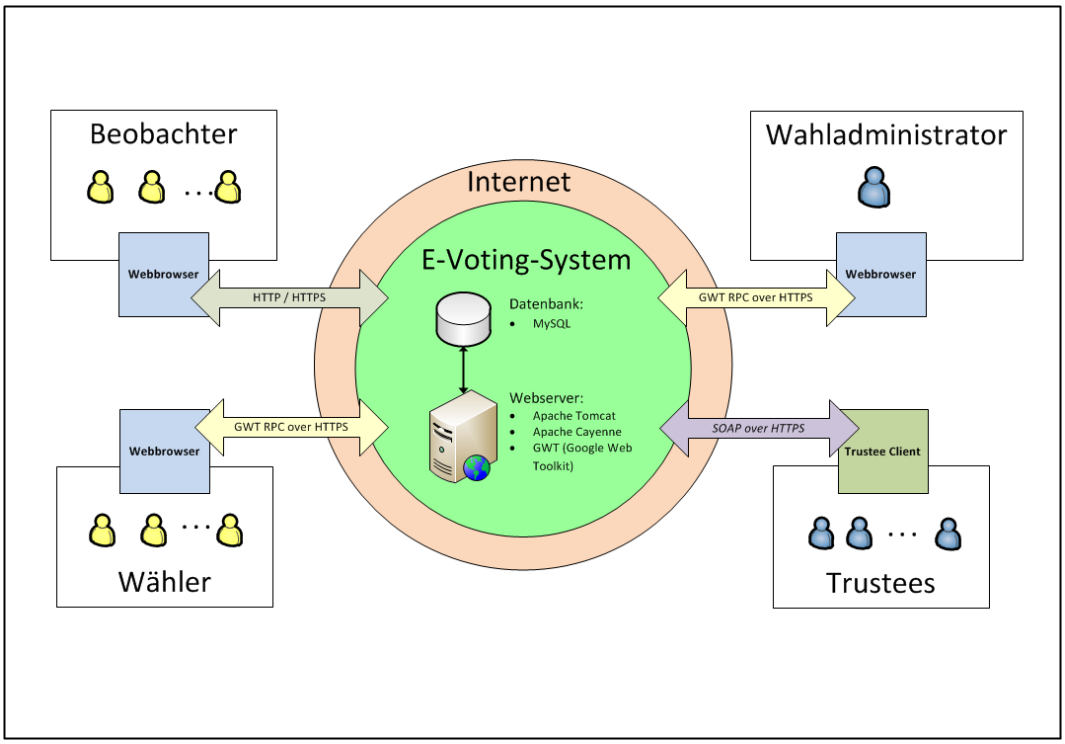

Abbildung 5.1.: Übersicht Systemarchitektur

5.2.1. Datenmodell

Um die zu einer Abstimmung gehörenden Daten persistent abzuspeichern, kommt eine relationale Datenbank, in unserem Fall MySQL, zum Einsatz. Die im Datenmodell enthaltenen Tabellen werden im Folgenden erklärt:

Tabelle “person”

Diese Tabelle beinhaltet eine Sammlung von Identitäten von Personen, die sich an einer Abstimmung beteiligen können. Dazu gehört eine E-Mail Adresse sowie ein X.509 Zertifikat, das alle weiteren Informationen enthält. Als weiteres Merkmal wird der SHA-1 Fingerprint des X.509 Zertifikats gespeichert. Dieser wird für die Authentifizierung des Benutzers am System verwendet. Da diese Tabelle als Pool von Identitäten für Administratoren, Trustees und Wählern verwendet wird muss diese Tabelle bei der Inbetriebnahme des Systems mit Daten gefüllt werden.

Tabelle “role”

Diese Tabelle beinhaltet die Rollen, die eine Person in “Comitia” einnehmen kann. Der Inhalt dieser Tabelle besteht aus drei statischen Einträgen, die die drei Rollen “Adminis-

trator", "Trustee" und "Voter" repräsentieren. Diese Tabelle muss bei der Inbetriebnahme des Systems mit Werten abgefüllt werden. Es ist nicht vorgesehen, den Inhalt dieser Tabelle zu verändern.

Tabelle "election"

Hier werden die Definitionen einer im System vorhandenen Abstimmung gespeichert. Ein Datensatz wird erstellt, wenn ein Administrator eine Abstimmung einrichtet (Use Case 1). Zu den Daten gehört die Fragestellung, der Status der Abstimmung (offen/-geschlossen) sowie der Public Key den die Wähler verwenden um ihre Stimme zu verschlüsseln. Nach Erstellung der Abstimmung ist dieses Feld noch leer, da dieser Public Key erst nach der Erstellung des Schlüsselmaterials durch die Trustees errechnet werden kann. Ein weiteres Merkmal ist der Threshold der Abstimmung, der definiert wie viele Trustees mindestens notwendig sind um das Resultat der Abstimmung zu entschlüsseln.

Tabelle "election_option"

Die in einer Abstimmung zur Verfügung stehenden Optionen oder Auswahlmöglichkeiten werden in dieser Tabelle repräsentiert. Nebst dem Text der die Option beschreibt, beinhaltet diese Tabelle auch Felder, die bei der Auszählung der Abstimmung verwendet werden. Dazu gehören die zwei Werte die beim homomorph aufsummieren der Stimmen dieser Option entstehen, sowie der aus der Entschlüsselung resultierende Klartext, welcher der Summe der *Stimmen* für diese Option entspricht. Jeder Datensatz in dieser Tabelle ist eindeutig einem Datensatz in der Tabelle "election" zugeordnet.

Tabelle "election_person"

Hier wird die Zuordnung zwischen Abstimmung, Rolle und Identität einer Person vorgenommen. Für jede Form von Beteiligung an einer Abstimmung wird hier ein Eintrag erstellt. Eine Person die beispielsweise eine Abstimmung erstellt und somit die Rolle des Administrators innehat und sich zudem als Wähler an der Abstimmung beteiligen will, erhält zwei Einträge in dieser Tabelle. In dieser Tabelle werden ausserdem Werte gespeichert, die im Verlaufe einer Abstimmung entstehen und je nach Rolle gefüllt werden oder leer bleiben. Dies sind für einen Wähler der Zeitstempel der Abstimmungszeit sowie acht Werte eines Proof of Validity der beweist, dass der Wähler tatsächlich nur eine Option ausgewählt hat. Für einen Trustee wird zusätzlich noch eine Nummer, sowie seinen Teil des Public Keys gespeichert. Die Datensätze werden bei der Erstellung der Abstimmung durch den Administrator erstellt.

Tabelle “secretexchange”

Bei der verteilten Schlüsselgenerierung müssen Trustees die Möglichkeit haben, Teile von Schlüsseln über einen sicheren Kanal (Vertraulichkeit, Integrität, Authentizität) austauschen zu können. Über die Tabelle “secretexchange” können solche Messages von Trustee zu Trustee gesendet werden. Nach Erhalt dieser Message überprüft der Empfänger seine Message und kann gegebenenfalls einen signierten “Complaint” gegen den Empfänger veröffentlichen. Dieser Complaint wird ebenfalls in dieser Tabelle abgespeichert.

Tabelle “ballot”

In diesem Bereich werden die einzelnen abgegebenen Stimmen in verschlüsselter Form gespeichert. Jeder “Ballot” ist eindeutig einem Voter (election_person) und einer Option (election_option) zugeordnet. Bei einer Stimmabgabe werden immer so viele Ballots erstellt wie Optionen zur Auswahl stehen. Stimmt ein Wähler in einer Abstimmung mit drei Optionen zur Auswahl ab, werden in dieser Tabelle drei Datensätze angelegt. Zusätzlich zum ElGamal Ciphertext, der aus zwei Werten besteht, wird zu jedem Ballot ein Proof of Validity gespeichert. Dieser beweist, dass der Wähler auch wirklich die Werten 0 (*Nein*) oder 1 (*Ja*) verschlüsselt hat.

Tabelle “trustee_result”

Nachdem alle Wähler ihre Stimme abgegeben haben, muss das Resultat homomorph aufsummiert und anschliessend dezentral durch die Trustees entschlüsselt werden. Dafür müssen die Trustees eine Teilentschlüsselung des Resultats durchführen. Das Resultat dieser Teilentschlüsselung wird in dieser Tabelle gespeichert. Auch hier muss der Trustee mit einem Zero-Knowledge-Beweis beweisen, dass er diesen Schritt korrekt durchgeführt hat. Die Werte von diesem Proof werden auch in dieser Tabelle gespeichert.

Verwendete Datentypen

Um sichere Kryptographie zu betreiben muss mit sehr hohen Zahlen (bei einer Parameterlänge von 1024 Bits sind es 309 Dezimalstellen) gerechnet werden. MySQL bietet keine Möglichkeit, so grosse Zahlen numerisch in der Datenbank abzulegen. Somit müssen solche Zahlen als Zeichenkette abgespeichert werden wofür sich der MySQL Datentyp TEXT gut eignet. Dies ist insofern unproblematisch da in Java der Datentyp “BigInteger” nur mit einer Zeichenkette die die eigentliche Zahl repräsentiert initialisiert werden kann.

Ansonsten wurden gängige Datentypen wie INT für kleinere Zahlen, Primärschlüssel und Fremdschlüssel, VARCHAR für kurze Zeichenketten sowie TINYINT für Booleanwerte verwendet.

Datencodierung

Bei der Speicherung von Bitfolgen wie sie beispielsweise bei der Erstellung von Signaturen entstehen, muss man sich Gedanken machen in welcher Form man diese abspeichert. Eine Möglichkeit ist die BASE64 Codierung, welche die Bitfolgen als ASCII Zeichen codiert und somit als Zeichenfolge repräsentierbar macht. So kann eine solche Bitfolge einfach in die Datenbank gespeichert werden. Für dieses Projekt wurde entsprechend BASE64 gewählt. Solche BASE64 codierte Zeichenfolgen sind in der Tabelle "secretexchange" zu finden.

Hashwerte werden ebenfalls als Zeichenfolge gespeichert, jedoch in hexadezimaler Form. Für Hashes ist dies die gängigste Form, somit arbeiten auch Tools wie z.B. "sha1sum" mit hexadezimalen Zeichenfolgen.

Die X.509 Zertifikate werden ohne Veränderungen im gängigen PEM Format in der Datenbank abgelegt.

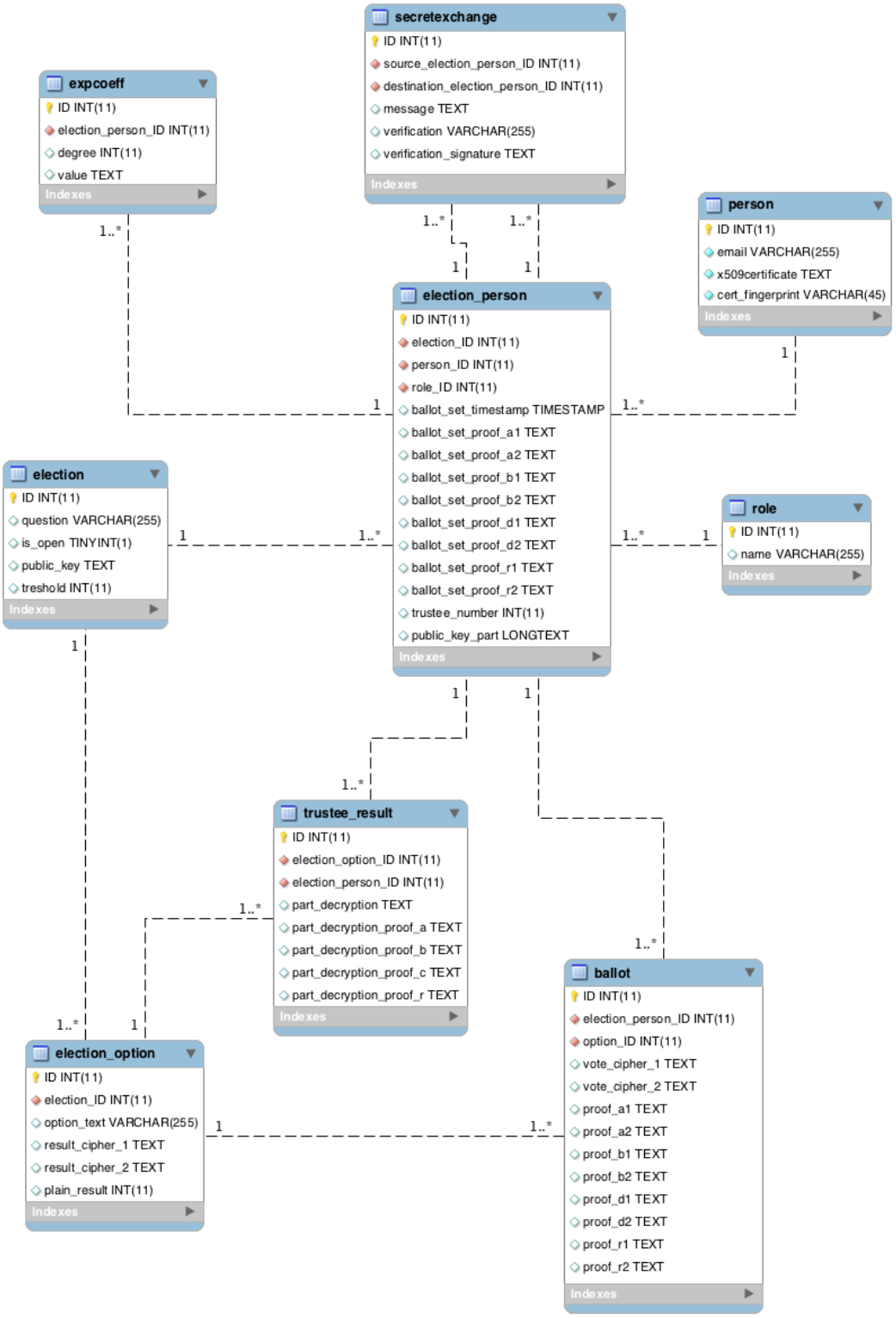

Abbildung 5.2.: Comitia Datenbank Modell

5.2.2. ORM Mapping mit Cayenne

Einfach gesagt lassen sich Datensätze in den einzelnen Tabellen in Objekte einer objektorientierten Programmiersprache abbilden. Jeder Datensatz in einer Tabelle entspricht einem Objekt, mit dem in der Applikationslogik gearbeitet werden kann. Die Aufgabe, dieses sogenannte "Mapping" durchzuführen, wird durch ein ORM (Object Relational Mapping) Framework übernommen. In diesem Projekt wurde als solches ORM Framework "Cayenne" vom Apache Projekt gewählt. Cayenne muss mitgeteilt werden, welche Entitäten (Tabellen) es in welche Klassen mappen soll. Dies kann einfach in einem Cayenne Hilfsprogramm namens "Cayenne Modeler" gemacht werden. Dieses Tool erstellt aufgrund dessen Mappings die nötigen Java Klassen, mit denen danach gearbeitet werden kann. Um die Persistenz der Instanzen dieser Klassen kümmert sich im Betrieb dann Cayenne, indem es die jeweiligen Operationen in SQL Code übersetzt und diese auf der Datenbank ausführt.

5.2.3. Das DTO Konzept

Cayenne macht es sehr einfach, Objekte einer bestimmten Entität persistent zu machen, indem einfach Java Objekte erstellt, gelöscht oder bearbeitet werden. Diese Javaobjekte haben jedoch den Nachteil, dass sie eine grosse Klassenhierarchie im Schlepptau haben. Will man nun ein solches Objekt mittels Serialisierung (z.B. über das Netzwerk zu einem Client übertragen) muss die gesamte Klassenhierarchie auch auf dem Client vorhanden sein, damit das Objekt korrekt deserialisiert werden kann. Das in unserem Projekt verwendete GWT bietet für die Clientseite nur einen begrenzten Teil von Klassen, zu denen Cayenne nicht zählt. Somit musste eine Objektstruktur gefunden werden, welche einfachere Objekte über das Netzwerk überträgt. Hier kommt das DTO (Data Transfer Object) Konzept ins Spiel. Eine DTO Klasse beinhaltet Attribute sämtlicher Nutzdaten entsprechend den Attributen in der ORM Klasse (oder der zugehörigen Tabelle). In unserem Fall gibt es für die Entität "Role" beispielsweise eine ORM Klasse "Role" mit der gesammten Vererbungshierarchie und den Attributen "ID" und "Name". Zusätzlich gibt es nun eine Klasse "RoleDTO", die ebenfalls die Attribute "ID" und "Name" hat und zudem serialisierbar ist. Für die Manipulation der Variablenwerte stehen entsprechende Getter- und Settermethoden zur Verfügung. Objekte in dieser Form können nun problemlos serialisiert und somit über das Netzwerk übertragen werden.

5.2.4. Beschreibung der Java Packages

Die Programmiersprache Java bietet die Möglichkeit, Code über so genannte Packages zu strukturieren. So wurde der Code von Comitia nach Themengebiet strukturiert um die Übersichtlichkeit zu wahren. Im Folgenden werden die einzelnen Packages und deren Inhalt kurz beschrieben.

Comitia

Package:	ch.bfh.ti.comitia
Beschreibung:	Dieses Package beinhaltet die GWT Moduldefinition. Diese Moduldefinition besteht aus einer XML Datei, die gewisse Parameter für das GWT Projekt definiert.

Cayenne Persistent Auto

Package:	ch.bfh.ti.comitia.cayenne.persistent.auto
Beschreibung:	Dieses Package wird automatisch von Cayenne erstellt. Für jedes definierte Objekt wird in diesem Package eine abstrakte Klasse generiert, die gewisse Logik für das Mapping enthält. Diese Klassen sollten nicht verändert werden. Änderungen sollten nur von Cayenne selbst gemacht werden.

Cayenne Persistent

Package:	ch.bfh.ti.comitia.cayenne.persistent
Beschreibung:	Für jedes in Cayenne definierte Objekt gibt es hier eine Klasse, die standardmässig nur die entsprechende Klasse vom Package ch.bfh.ti.comitia.cayenne.persistent.auto erweitert. Die initialen Klassen wurden auch durch Cayenne generiert. Im Gegensatz zu den Klassen im Paket “auto” können diese Klassen mit eigener Logik erweitert werden.

Client

Package:	ch.bfh.ti.comitia.client
Beschreibung:	Dies ist ein GWT spezifisches Package und beinhaltet den Code, der vom GWT Compiler in JavaScript Code umgewandelt wird. Es beinhaltet somit jegliche clientseitige Logik der Webapplikation und somit auch die Logik des Verschlüsselungs- und Beweisvorgangs. Nachfolgend ein paar besonders erwähnenswerte Klassen: • **CastBallotDialog:** Die Klasse CastBallotDialog ist für die Darstellung und Logik des Dialogs für die Stimmabgabe verantwortlich. Sie beinhaltet somit einen essentiellen Teil der Applikationsfunktionalität, nämlich die Algorithmen zur Erstellung eines Ballots. • **Comitia:** Diese Klasse ist für die Funktionalität der Startseite verantwortlich. Sie ist ein so genannter "Entry Point" der GWT Applikation, also eine Art Main-Klasse, die dann wiederum gebrauch von anderem Code macht. • **CreateElectionDialog:** Diese Klasse beinhaltet die Definition und die Logik des Dialogs zur Erstellung von neuen Abstimmungen. • **ElectionAdminDialog:** In dieser Klasse befindet sich die Definition des Administrationsdialogs einer Election. • **ElectionResultDialog:** Beinhaltet den Dialog für die Anzeige des Resultats.

Client Exception

Package:	ch.bfh.ti.comitia.client.exception
Beschreibung:	Dieses Package beinhaltet eine projektspezifische Exception "ComitiaException", die im Fall eines Fehlers geworfen wird, beispielsweise wenn ein Wähler doppelt abstimmen will.

Client Thirdparty

Package:	ch.bfh.ti.comitia.client.thirdparty
Beschreibung:	Dieses Package beinhaltet von GWT Incubator extrahierte Klassen für die Fortschrittsanzeige des clientseitigen Verschlüsselung-Vorgangs. Incubator ist eine Sammlung von geplanten, aber noch nicht stabilen Funktionalitäten in GWT. Incubator wird von Google in dieser Form nicht mehr weiterentwickelt. Aus diesen Gründen wurden diese Klassen aus dem Paket extrahiert und in das Projekt integriert. Diese Klassen sind allesamt Fremdmaterial und nicht im Verlaufe dieses Projekts entstanden.

Server

Package:	ch.bfh.ti.comitia.server
Beschreibung:	Dieses Package enthält die Implementation der vom E-Voting-System zu Verfügung gestellten GWT-RPC Services. Diese werden gebraucht um Daten zwischen dem client- und dem serversseitigen Teil der Webapplikation auszutauschen. Folgende Klassen sind besonders erwähnenswert: • **BallotServiceImpl:** Diese Klasse implementiert die Funktionalität vom Ballot Service. Er nimmt vom Wähler erstellte Ballots entgegen und schreibt diese in die Datenbank. • **ElectionServiceImpl:** Enthält die Funktionalität für die Erstellung und die Administration von Abstimmungen. • **PersonServiceImpl:** Stellt die Funktionalität für personenbezügliche Abfragen bereit. Dieses Serverpackage ist von GWT strukturmässig vorgegeben.

Server Filter

Package:	ch.bfh.ti.comitia.server.filter
Beschreibung:	Dieses Paket enthält Filterklassen, durch welche jede Anfrage zum Server läuft. In einer solche Filterklasse wurde der Authentifizierungsmechanismus implementiert, der sicherstellt, dass nur berechtigte Personen diese Services aufrufen können.

Shared

Package:	ch.bfh.ti.comitia.shared
Beschreibung:	Dieses GWT spezifische Package beinhaltet Klassen, die sowohl auf der Client- wie auch auf der Serverseite der Webapplikation zur Verfügung stehen müssen. Dies sind beispielsweise sämtliche DTO Klassen, dessen Objekte zum Transport von Daten verwendet werden. Ebenfalls in diesem Package findet sich die Klasse “CryptoSettings”, die die domainspezifischen ElGamal Parameter enthält.

Sigma

Package:	ch.bfh.ti.comitia.sigma
Beschreibung:	In diesem Package finden sich Klassen, die im Zusammenhang mit den Zero-Knowledge-Beweisen stehen. So findet sich hier u.a. eine Klasse die für die Validierung der im System vorhandenen Zero-Knowledge-Beweisen verwendet wird.

Trustee Client

Package:	ch.bfh.ti.comitia.trusteeclient
Beschreibung:	Dieses Package beinhaltet alle Klassen, die für das Trustee Client GUI benötigt werden.

Util

Package:	ch.bfh.ti.comitia.util
Beschreibung:	Dieses Package enthält nützliche Werkzeuge, die bei der Entwicklung des Prototypen eingesetzt wurden.

Webservices

Package:	ch.bfh.ti.comitia.webservices
Beschreibung:	Dieses Package enthält die vom Server zu Verfügung gestellten SOAP Webservices. Beim Start von Tomcat wird mit Hilfe von JAX-WS eine Schnittstelle gemäss der Klassendefinition von der Klasse "ElectionWebService" gemacht. Dieser Webservice beinhaltet unter anderem Funktionalität für die Überprüfung von Zero-Knowledge-Beweisen, für das Tallying der Abstimmung und für die letzten Schritte der Entschlüsselung mittels Lagrange Interpolation.

5.2.5. Webapplikation

Ein grosser Teil von Comitia wurde in Form einer Webapplikation realisiert. Wie fast jede moderne Webapplikation hat Comitia Programmteile die auf dem Server ablaufen wie auch Teile die auf dem Client ausgeführt werden müssen. Aufgrund der Vertraulichkeit von Wählerstimmen müssen diese komplett auf dem Client des Wählers erstellt werden. Dies erfordert einige Programmlogik, die in Webapplikationen üblicherweise mit der Programmiersprache "JavaScript" abgebildet wird.

Für die Realisation dieses Projekts wurde ein Java Framework von Google namens "Google Web Toolkit" (GWT) eingesetzt. Mit GWT hat Google versucht, die Server- und Clientseitige Programmierung zu vereinen. Technisch gesehen übersetzt ein von Google zur Verfügung gestellter Compiler den erstellten Java Code in clientseitigen JavaScript Code. Dies hat den Vorteil, dass man eine moderne, interaktive Webapplikation komplett in Java realisieren und die Vorzüge der objektorientierten Programmierung auf die Clientseite ausdehnen kann.

Google möchte mit Hilfe dieses Toolkits auch vom Ansatz wegkommen, für jede Maske und Oberfläche die gesamte Seite neu zu laden. Stattdessen werden gezielt Daten über so genannte Services auf dem Server ausgetauscht. Dies hat unter anderem

den Vorteil, dass die Performance erheblich gesteigert werden kann, da dank gezieltem nachladen von Daten weniger Informationen zwischen Server und Client ausgetauscht werden müssen.

Um Daten zwischen Client- und Serverseite auszutauschen stellt GWT eine Technologie namens GWT-RPC (Google Web Toolkit Remote Procedure Call) zur Verfügung. Damit lassen sich solche Services sehr einfach implementieren. Für die Comitia Webapplikation wurden folgende Services implementiert:

Election Service

Dieser Service wird vom Clientteil der Applikation gebraucht, um neue Elections zu erstellen respektive diese aus dem System zu holen und zu administrieren. Der Election Service stellt folgende Methoden zur Verfügung:

closeElection

Methode:	closeElection()
Argumente:	ElectionDTO
Rückgabewert:	keine
Beschreibung:	Diese Methode schliesst eine Abstimmung, so dass keine weiteren Stimmen reinkommen können.

createElection

Methode:	createElection()
Argumente:	keine
Rückgabewert:	Liste vom Typ ElectionDTO
Beschreibung:	Diese Methode erstellt eine neue Abstimmung gemäss dem als Argument übergebenen Objekts.

getAdminElections

Methode:	createElection()
Argumente:	ElectionDTO
Rückgabewert:	Liste vom Typ ElectionDTO
Beschreibung:	Diese Methode gibt alle Abstimmungen in dem der Aufrufer Administrator ist zurück

getElections

Methode:	getElections()
Argumente:	keine
Rückgabewert:	Liste vom Typ ElectionDTO
Beschreibung:	Diese Methode gibt alle definierten Abstimmungen im System zurück.

getTrusteeElections

Methode:	getTrusteeElections()
Argumente:	keine
Rückgabewert:	Liste vom Typ ElectionDTO
Beschreibung:	Diese Methode gibt alle Abstimmungen in denen der Aufrufer als Trustee definiert ist zurück.

getVoteElections

Methode:	getVoteElections()
Argumente:	keine
Rückgabewert:	Liste vom Typ ElectionDTO
Beschreibung:	Diese Methode gibt alle Abstimmungen in denen der Aufrufer als Wähler definiert ist zurück.

openElection

Methode:	openElection()
Argumente:	ElectionDTO
Rückgabewert:	keine
Beschreibung:	Diese Methode rechnet den Public Key für die Abstimmung und eröffnet die Abstimmung.

tallyElection

Methode:	tallyElection()
Argumente:	ElectionDTO
Rückgabewert:	keine
Beschreibung:	Diese Methode erteilt dem Server den Befehl die Resultate homomorph aufzusummieren.

Ballot Service

Dieser Service wird gebraucht um verschlüsselte Stimmen von Wählern entgegen zu nehmen und entsprechend in der Datenbank abzulegen. Grundsätzlich wäre es möglich

gewesen, die Methode dieses Services in den Election Service einzubetten. Als einziger Service wird dieser von Wählern aufgerufen und ist somit extrem exponiert. Die separate Verarbeitung von Anfragen ermöglicht die Implementation von speziellen Sicherheitsmechanismen, beispielsweise IP Filtering auf dem Webserver.

Der Ballot Service stellt eine Methode mit folgender Definition zur Verfügung:

castBallotSet

Methode:	castBallotSet()
Argumente:	Liste vom Typ BallotDTO, ElectionDTO, BallotSetProofDTO
Rückgabewert:	keine
Beschreibung:	Diese Methode nimmt die Stimme eines Wählers entgegen. Diese Besteht aus einer Liste von BallotDTO Objekten, wobei die Grösse der Liste der Anzahl der wählbaren Optionen entspricht. Das ElectionDTO Objekt wird gebraucht, um die Zuordnung zur Abstimmung zu machen. im BallotSetProofDTO Objekt werden die Werte für den ZKP des Ballotsets übertragen.

Person Service

Der Person Service ist ein Hilfs-Service um den Clientteil mit Informationen zu Personen zu Versorgen. Mit ihm lässt sich beispielweise ein PersonDTO Objekt aufgrund einer E-Mail Adresse anfordern.

getPersonByEmail

Methode:	getPersonByEmail()
Argumente:	E-Mail Adresse als String
Rückgabewert:	das entsprechende PersonDTO
Beschreibung:	Diese Methode ermöglicht es, eine Person aufgrund einer E-Mail Adresse im Identitäten Pool (Tabelle person) zu suchen.

getPersonSuggestions

Methode:	getPersonSuggestions()
Argumente:	ein Suchpattern als String
Rückgabewert:	eine Liste vom Typ String mit den Suchresultaten
Beschreibung:	Diese Funktionalität wird gebraucht um bei der Erfassung von Abstimmungen Vorschläge von Personen zusammenzustellen.

Log Service

Dieser Infrastrukturservice ermöglicht es, Meldungen die vom Clientteil produziert werden an den Serverteil zu senden, der dies dann in sein Logfile schreibt. Er kann auch für Debuggingzwecke gebraucht werden. Der Service bietet pro Logstufe (info, warn, error, fatal, debug und trace) eine Methode, der ein String übergeben werden kann.

5.2.6. Trustee Client

Im CGS97 Protokoll gibt es für die Interaktion der Trustees mit dem E-Voting-System einige Vorgaben, die sich mit einer Webapplikation nicht oder nur mit markanten Einschränkungen realisieren lassen. Die Schwierigkeit ist der Austausch von Nachrichten unter den Trustees über einen sicheren Kanal und den damit verbundenen kryptographischen Anforderungen. Aus diesem Grund wurde für die Trustees ein Rich Client in Java realisiert. Auf diese Art und Weise ist es möglich, auf dem Clientsystem abgelegtes Schlüsselmaterial (Client Zertifikat mit zugehörigem Private Key) zu verwenden und somit asymmetrische Kryptoverfahren für eine sichere und vertrauliche Kommunikation unter den Trustees zu ermöglichen.

Als Kommunikationstechnologie zwischen Trustee Client und dem E-Voting-System wird SOAP over HTTPS verwendet. Das zentrale E-Voting-System stellt mittels Webservices die benötigte Kommunikationsschnittstellen mit dem Comitia Webserver zu Verfügung. Dieser Kommunikationskanal ist gleich wie beim Web Teil über HTTPS gesichert. Die Implementation dieser Webservices wird durch die JAX-WS Technologie unterstützt. Folgende Services stehen für die Kommunikation zwischen Trustee Client und dem E-Voting-System zu Verfügung:

Election Web Service

Der Election Web Service ist die einzige Kommunikationsschnittstelle zwischen Trustee Clients und dem E-Voting-System. Er bietet alle Services, damit die Voraussetzungen für die Eröffnung einer Abstimmung und Entschlüsselung des Abstimmungsresultats geschaffen beziehungsweise durchgeführt werden können.

getDstSecretMsgVerificationData

Methode:	getDstSecretMsgVerificationData()
Argumente:	ElectionDTO, PersonDTO
Rückgabewert:	eine Liste von Arrays des Typs String
Beschreibung:	Diese Methode gibt alle Complaints und die dazugehörigen Signaturen zurück, die ein Trustee erhalten hat.

getElections

Methode:	getElections()
Argumente:	keine
Rückgabewert:	eine Liste vom Typ ElectionDTO
Beschreibung:	Diese Methode gibt alle Abstimmungen zurück.

getExpCoeffs

Methode:	getExpCoeffs()
Argumente:	ElectionDTO
Rückgabewert:	eine Liste des Typs ExpcoeffDTO
Beschreibung:	Diese Methode gibt alle Koeffizienten einer Abstimmung zurück.

getExpCoeffsOfTrustee

Methode:	getExpCoeffsOfTrustee()
Argumente:	ElectionDTO, PersonDTO
Rückgabewert:	eine Liste des Typs ExpcoeffDTO
Beschreibung:	Diese Methode gibt alle Koeffizienten eines Trustees einer Abstimmung zurück.

getMyPublishedSecretMessages

Methode:	getMyPublishedSecretMessages()
Argumente:	ElectionDTO, PersonDTO
Rückgabewert:	eine Liste vom Typ SecretexchangeDTO
Beschreibung:	Diese Methode gibt alle gesendeten Secret Messages des authentifizierten Trustees für eine Abstimmung zurück.

getSecretMessages

Methode:	getSecretMessages()
Argumente:	ElectionDTO
Rückgabewert:	eine Liste vom Typ SecretexchangeDTO
Beschreibung:	Diese Methode gibt alle erhaltenen Secret Messages des authentifizierten Trustees für eine Abstimmung zurück.

getSrcSecretMsgVerificationData

Methode:	getSrcSecretMsgVerificationData()
Argumente:	ElectionDTO, PersonDTO
Rückgabewert:	eine Liste von Arrays des Typs String
Beschreibung:	Diese Methode gibt alle Complaints und die dazugehörigen Signaturen zurück, die ein Trustee gesendet hat.

getTrusteesOfElection

Methode:	getTrusteesOfElection()
Argumente:	keine
Rückgabewert:	eine Liste vom Typ ElectionDTO
Beschreibung:	Diese Methode gibt alle Abstimmungen, in dem der Aufrufer als Trustee definiert ist, zurück.

hasParticipatedDec

Methode:	hasParticipatedDec()
Argumente:	ElectionDTO, PersonDTO
Rückgabewert:	boolean
Beschreibung:	Diese Methode gibt zurück, ob ein Trustee an der Entschlüsselung des Abstimmungsresultats teilgenommen hat (true) oder nicht (false).

isElectionTallied

Methode:	isElectionTallied()
Argumente:	ElectionDTO
Rückgabewert:	boolean
Beschreibung:	Diese Methode gibt zurück, ob eine Abstimmung ausgezählt wurde (true) oder nicht (false).

sendExpCoeffs

Methode:	sendExpCoeffs()
Argumente:	SecretexchangeDTO, ElectionDTO
Rückgabewert:	keine
Beschreibung:	Diese Methode nimmt eine Liste von Koeffizienten des authentifizierten Trustees entgegen und speichert sie für die Abstimmung ab.

sendPublicKeyPart

Methode:	sendPublicKeyPart()
Argumente:	ein publicKeyPart des Typs Biginteger, ElectionDTO
Rückgabewert:	keine
Beschreibung:	Diese Methode speichert den Teilschlüssel des authentifizierten Trustees in der Datenbank.

sendSecretMessage

Methode:	sendSecretMessage()
Argumente:	SecretexchangeDTO
Rückgabewert:	keine
Beschreibung:	Diese Methode nimmt die Secret Messages des authentifizierten Trustees entgegen und speichert sie für die Trustees, die an der selben Abstimmung teilnehmen, in der Datenbank ab.

sendTrusteeResults

Methode:	sendTrusteeResults()
Argumente:	eine Liste des Typs TrusteeResultDTO, ElectionDTO
Rückgabewert:	keine
Beschreibung:	Diese Methode speichert die Trustee results des authentifizierten Trustees in der Datenbank.

setComplaint

Methode:	setComplaint()
Argumente:	SecretexchangeDTO
Rückgabewert:	keine
Beschreibung:	Diese Methode publiziert einen Complaint auf eine Secretexchange Message.

5.2.7. Authentifikation & Authorisation

Um Manipulationen in Abstimmungen zu vermeiden, ist es für ein E-Voting-System wichtig, das es Identitäten von Benutzern sowie deren Berechtigungen berücksichtigt und entsprechend handelt. Konkret geht es in diesem System darum, die einzelnen oben beschriebenen Services so abzusichern, dass sie nur von berechtigten Personen aufgerufen werden können. In diesem Projekt haben wir uns dafür entschieden, dieses Problem mit Hilfe von X.509 Zertifikaten zu lösen. Folgende Gründe haben zu dieser Entscheidung geführt:

- X.509 Zertifikate haben gleich wie Abstimmungen einen offiziellen Charakter
- Abstimmungen werden oft innerhalb von Organisationen durchgeführt, welche mehr und mehr über PKIs verfügen
- Mit SuisseID wurde in der Schweiz im Jahr 2010 die Weiche für diese Technologie in der Schweiz gestellt
- Diese Zertifikate bieten Schlüsselmaterial für eine äussert sichere Kommunikation auf Basis von asymmetrischen oder hybriden Kryptographie
- Applikationsseitig ist keine Passwortverwaltung notwendig

Die Überprüfung dieser Clientzertifikate, die von den beteiligten Personen zusammen mit ihrem Private Key im Browser hinterlegen müssen, wird vom Webserver als Teil vom SSL/TLS Handshake durchgeführt. Der Webserver (in diesem Fall Apache Tomcat) muss so konfiguriert werden, dass er

- Verbindungen über HTTPS (HTTP gesichert mit SSL/TLS) zulässt
- Beim SSL/TLS Handshake vom Client ein Clientzertifikat anfordert und dieses verifiziert

Eine Anleitung wie der Tomcat konfiguriert werden muss, findet sich in der Installationsanleitung im Anhang dieses Dokuments A.3.2.

Das Zertifikat wird anschliessend an einen Filter weitergeleitet, den jeder HTTP Request durchlaufen muss. Dieser ordnet das enthaltene Zertifikat einer Identität in der Datenbanktabelle “person” zu. So wird sichergestellt, dass Operationen nur von berechtigten Personen durchgeführt werden können.

5.3. Benutzerfreundlichkeit

Eine Stimmabgabe sollte für den Wähler möglichst rasch, einfach und benutzerfreundlich von statten gehen. Für die Teilnahme an einer Abstimmung muss keine zusätzliche Software installiert werden. Jegliche Interaktion mit dem E-Voting-System ist über einen Standard Webbrowser möglich.

M. Fatih Karayumak [9] hat in seiner Bachelor-Thesis die Benutzerfreundlichkeit von HELIOS V. 3.0 analysiert. Folgende Verbesserungsvorschläge haben wir aus seiner Arbeit in unser Produkt miteinbezogen:

- Bevor ein Wähler die gewählte Option bestätigt und verschlüsselt an den E-Voting Webserver sendet, kann er zur Auswahl der Optionen zurückkehren, ohne den Wahlvorgang erneut beginnen zu müssen.
- Für die Auswahl der Optionen wurden Radio Buttons und keine Checkboxen verwendet, so können fehlerhafte Eingaben des Wählers einfacher korrigiert werden.
- Ein besonderes Augenmerk wurde auf die Verständlichkeit von Bezeichnungen und Begriffen gelegt.
- Einem Wähler wird immer nur das nötigste an Informationen angezeigt, er kann selber entscheiden, ob er die kryptographischen Werte ansehen will oder nicht.

6. Fazit

Das in dieser Bachelorarbeit entstandene Produkt stellt eine Plattform für kryptographisch sichere, nachvollziehbare und vertrauliche Abstimmungen zu Verfügung. Es bestätigt in erster Linie die Durchführbarkeit der Implementation eines voll funktionstüchtigen E-Voting-Systems nach CGS97 [1] und zeigt auf, wie und mit welchen Mitteln es realisiert werden kann.

6.1. Ergebnisse

Das E-Voting-System ist für den Wähler so einfach wie möglich gehalten. Die Stimmabgabe, sowie Erstellung und Administration einer Abstimmung kann vollumfänglich mittels Webapplikation durchgeführt werden. Für die verteilte Schlüsselgenerierung und Entschlüsselung musste aufgrund mathematisch aufwendigen Rechenoperationen und dem benötigten Zugriff auf die Private Keys der Trustees allerdings eigens ein Java Rich Client entwickelt werden. Dieser kann ebenfalls aus dem Webbrowser gestartet werden. Um die Vertraulichkeit, Datenintegrität und Authentizität im E-Voting-System zu jedem Zeitpunkt einer Abstimmung gewährlisten zu können, wurden ausschliesslich öffentliche, kryptographisch sichere Methoden angewendet und implementiert. Eine grosse Herausforderung unserer Bachelorarbeit war die Umsetzung der bereits erwähnten kryptographischen Bausteine in Java, genauer gesagt die Vereinigung dieser Elemente in ein voll funktionales Gesamtsystem, welches die geforderten Ansprüche eines transparenten und sicheren E-Voting-Systems ausnahmslos einhält. Mit ein wenig Stolz können wir behaupten, dass uns dies auch gelungen ist.

6.2. Ausblick

Es sind einige Ideen vorhanden, mit welchen das Produkt erweitert werden kann. Zum Beispiel gibt es Methoden nach Jörn Schweisgut [10], wie die Problematik der Quittungsfreiheit und Erpressungsresistenz mit Observern entschärft werden könnte. Eine berechtigte Frage ist auch, wie eine höhere Akzeptanz der Wähler für elektronische Wahlen erreicht werden soll, wenn mit den aus Anwendersicht komplizierten digitalen

Zertifikaten gearbeitet wird. Für Abstimmungen ohne offiziellen Charakter wäre durchaus denkbar, die Identifikation eines Wählers mit Benutzername und Passwort durchzuführen oder bekannte Single-Sign-On-Methoden wie OpenID, Facebook Connect oder Twitter Accounts anzuwenden. Die Entscheidung mit Zertifikaten zu arbeiten, öffnet andererseits das Spektrum der Einsatzmöglichkeiten für elektronische Abstimmungen. Beispielsweise könnte man mittels offiziell beglaubigtem elektronischen Zertifikat (z.B. SuisseID) staatliche Wahlen durchführen. Durch eine automatische Auszählung der Stimmen könnte viel Zeit und Kosten eingespart werden.

Die heutige Generation der Browser und deren JavaScript-Engines lassen bezüglich der Performanz noch zu wünschen übrig, so dauert die Verschlüsselung einer Stimme mit der momentan als sicher eingestuften Schlüssellänge von 1024 Bits in Firefox 4 (Beta 9) auf einem zeitgemässen Rechner immer noch einige Minuten. Kommende Browser-Generation werden den Einsatz grösserer Schlüssellängen erlauben, was zu einer schnelleren Stimmabgabe führen wird.

Interessant wäre auch die Kopplung an ein sicheres Bulletinboard, zum Beispiel gemäss Peters [11].

Eine nützliche Erweiterung wäre auch, dass bei der Eröffnung oder Auszählung einer Abstimmung die teilnehmenden Wähler entsprechend informiert werden. Ebenfalls sollten die Trustees bei Statusänderungen einer Abstimmung davon in Kenntnis gesetzt werden.

6.3. Danksagung

An dieser Stelle möchten wir uns bei allen beteiligten Personen bedanken, die uns unterstützt und die Vollendung unserer Bachelor-Thesis ermöglicht haben.

Ein besonderes Dankeschön gilt unseren Betreuern Prof. Dr. Rolf Haenni und Prof. Reto Koenig für die stetige Hilfsbereitschaft, die zahlreichen konstruktiven Gespräche, die Bereitschaft und der Wille uns in jeglicher Form zu unterstützen. Dies war essentiell um die Hürden auf dem Weg zum erfolgreichen Abschluss unserer Thesis meistern zu können.

Unserem Experten Prof. Dr. Andreas Spichiger möchten wir für das bereichernde und aufschlussreiche Gespräch danken. Es hat uns aufgezeigt, dass wir in die richtige Richtung zielen. Zusätzlich haben die Verbesserungsvorschläge und daraus resultierenden Massnahmen einen wesentlichen Anteil zu unserem Endprodukt beigetragen.

Dank möchten wir auch den kompetenten Dozenten der Berner Fachhochschule aussprechen, insbesondere den Dozenten aus dem Vertiefungsrichtung IT-Security, die unsere Begeisterung an diesem Thema geweckt haben.

Anhänge

A. Installationsanleitung

Die folgende Anleitung soll es ermöglichen, das in dieser Arbeit beschriebene Softwareprodukt erfolgreich in Betrieb zu nehmen. Grundlegende Kenntnisse im Umgang mit Webserver, Datenbankserver sowie Java werden vorausgesetzt. Ebenfalls hilfreich sind Grundlagen im Bereich von Zertifikaten.

A.1. Zertifikate

Dieses Softwareprodukt benötigt einige SSL-Zertifikate, zum Beispiel für die verschlüsselte HTTP-Verbindung zwischen dem Browser des Wählers und dem Webserver. Die Zertifikate werden zudem gebraucht, um einen Benutzer auf der Webapplikation zu identifizieren. Des Weiteren werden die Zertifikate bzw. die zugehörigen Private Keys für das Verschlüsseln von gewissen Werten, die die Trustees untereinander austauschen, verwendet. Alle diese Zertifikate sollten normalerweise von einer anerkannten Certification Authority (CA) ausgestellt werden. Zu Testzwecken hat die Projektgruppe eine eigene CA namens “Comitia Root CA” erstellt und folgende Sub-CAs kreiert:

- Comitia Personal CA1: Für das Signieren von Daten und der Überprüfung der Signatur
- Comitia Personal CA2: Für das Verschlüsseln und Entschlüsseln von Daten
- Comitia Personal CA3: Für die TLS Web-Client Authentifizierung
- Comitia Server CA: Für das SSL-Zertifikat des Webservers

Die CAs wurden mit Hilfe des OpenSource-Tools TinyCA2 erstellt. Die benötigten Dateien, um mit TinyCA2 zu arbeiten sowie alle erstellten Zertifikate und die verwendeten Passwörter befinden sich auf der Software-DVD.

Die nachfolgenden Screenshots sollen helfen, im Bedarfsfall ein neues Zertifikat für einen weiteren Benutzer anzulegen. Dazu öffnet man nach dem Start von TinyCA2 die CA “Comitia Personal CA3”, wonach eine Übersicht der existierenden Zertifikate angezeigt wird.

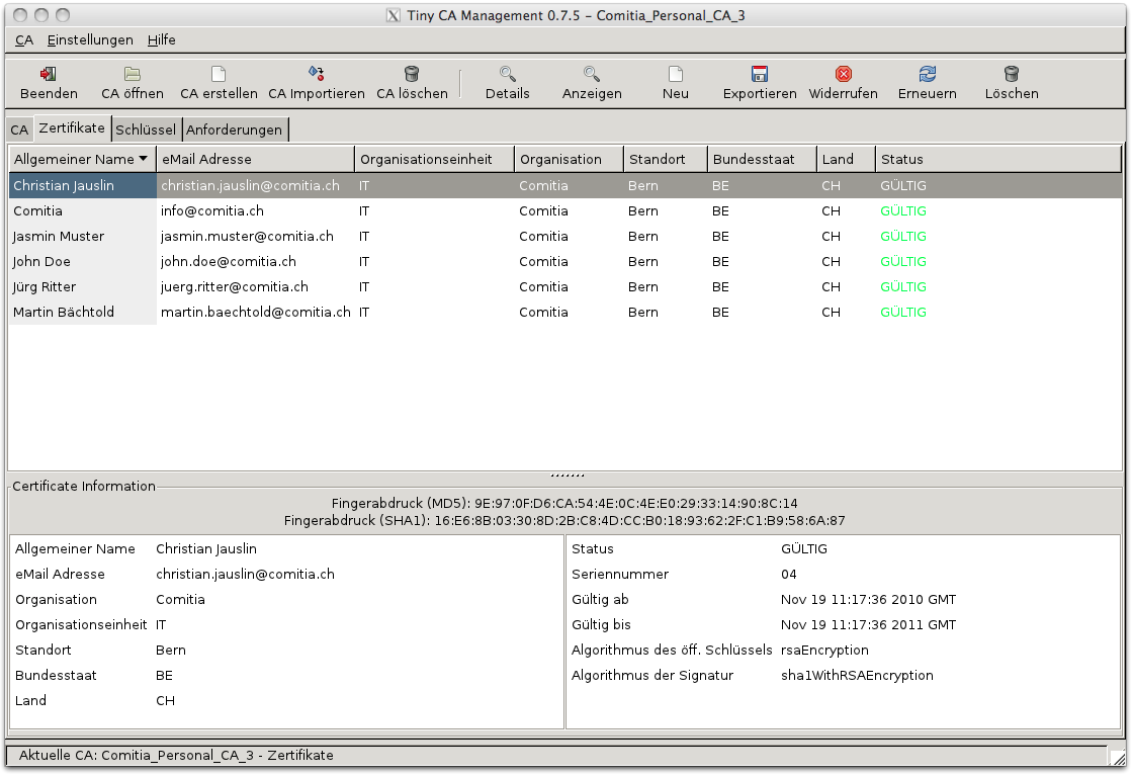

Abbildung A.1.: CA Übersicht

Über den Knopf “Neu” kann ein neues Benutzer-Zertifikat angelegt werden.

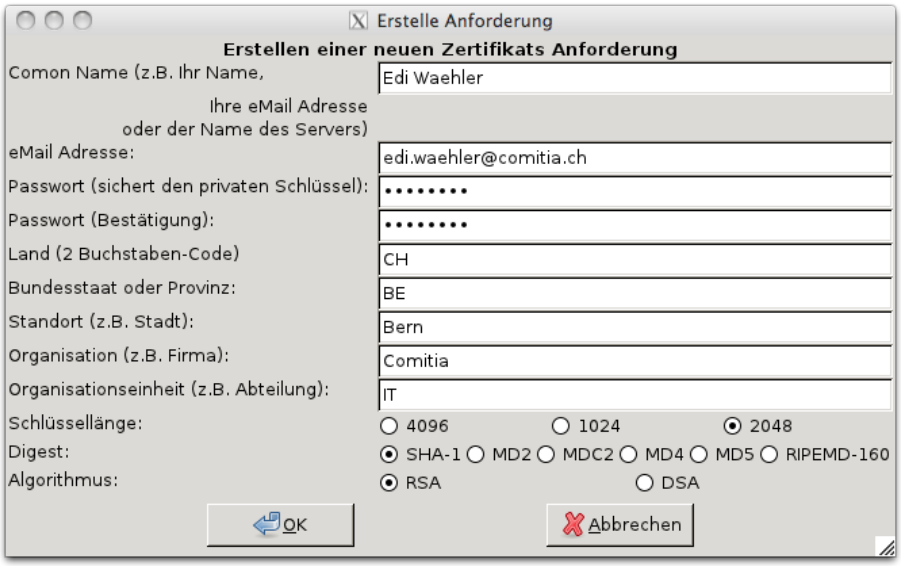

Abbildung A.2.: Zertifikat erstellen 1/2

Das Passwort der “Comitia Personal CA3” muss bekannt sein. Es befindet sich in der README-Datei auf dem beiliegenden Datenträger. Für die TLS Web client authentication muss die OID “1.3.6.1.5.5.7.3.2” eingetragen werden.

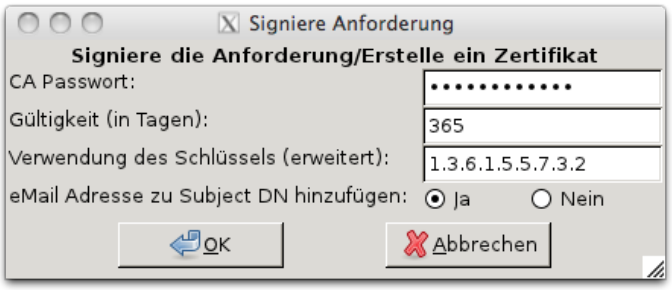

Abbildung A.3.: Zertifikat erstellen 2/2

Über die Schaltfläche “Exportieren” kann das Zertifikat und der Private Key in diversen Formaten exportiert werden. Hierbei muss ein Export-Passwort gewählt werden. Ebenfalls muss das bei der Erstellung des Zertifikats verwendete Passwort bekannt sein.

Abbildung A.4.: Zertifikat exportieren

Nun kann dieses Zertifikat im Browser des Teilnehmers installiert werden.

Zudem muss der Benutzer in der Datenbank erfasst, indem die E-Mail-Adresse des Benutzers, des Zertifikats und der SHA1-Fingerprint des Zertifikats in einem neuen Datensatz eingetragen wird.

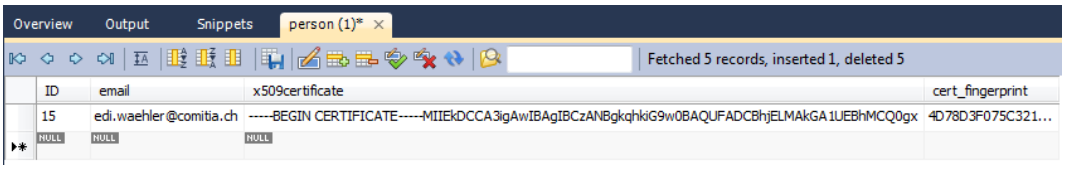

Abbildung A.5.: Benutzer in Datenbank eintragen

Das Zertifikat und den Fingerprint erhält man, wenn man im TinyCA2 auf den Knopf "Exportieren" drückt und das Format "PEM" auswählt und dabei die Option "Fingerabdruck hinzufügen" aktiviert.

Abbildung A.6.: PEM exportieren

Listing A.1: Zertifikat mit Fingerprint (PEM-Format)

```
Fingerprint (MD5): E9:E6:83:B2:83:DE:4D:3A:C4:93:88:45:95:2A:4B:49
Fingerprint (SHA1): 4D:78:D3:F0:75:C3:21:2A:08:C3:AE:5B:55:B2:F3:8C:BF:DF:0A:31

-----BEGIN CERTIFICATE-----
MIIEkDCCA3igAwIBAgIBCzANBgkqhkiG9w0BAQUFADCBhjELMAkGA1UEBhMCQ0gx
CzAJBgNVBAgTAkJFMQ0wCwYDVQQHEwRCZXJuMRAwDgYDVQQKEwdDb21pdGlhMQsw
CQYDVQQLEwJJVDEeMBwGA1UEAxMVQ29taXRpYSBQZXJzb25hbCBDQSAzMRwwGgYJ
KoZIhvcNAQkBFg1jYUBjb21pdGlhLmNoMB4XDTExMDEyMDA4MjY1NVoXDTEyMDEy
MDA4MjY1NVowgYUxCzAJBgNVBAYTAkNIMQswCQYDVQQIEwJCRTENMAsGA1UEBxME
[...]
VGutZdohkKCd3PO+l3yIOfi1oqFsw4k2N5HJxui1WMSTfBcOFZEQ1PUh3PzZ6HFd
3cDsAXLOgRPLIRIcg886u9ktz/oXg5Lilu6CVgcD24vb2lL2mEbmnkVJuEbsPinn
QSuVT1hgzVwqbpVINAD8wjbYcMiUOr7uBVHd/YgGenEw/KAy4vdr33oMazon/VJc
y4EWThvQgoUFfFkPv+gsTwbIiF+C1sZOdSTwHGRfag0tMle7+OKvCirPBLfJnbJz
EsHlOoQp/Ta7ZdLhDdKnuJgZEBk=
-----END CERTIFICATE-----
```

A.2. Benötigte Software

Für die Inbetriebnahme des Produkts müssen auf dem vorgesehenen Server einige Grundkomponenten installiert und lauffähig sein. Dies sind:

- **Java:** Java sollte mindestens in der Version 1.5.0 Standard Edition auf dem Rechner installiert sein. Java kann kostenlos für verschiedene Plattformen unter folgenden Adresse bezogen werden: `http://www.oracle.com/java/`
- **MySQL:** Um Daten persistent abzuspeichern, wurde in diesem Projekt das relationale Datenbanksystem MySQL gewählt. Installationspakete und Installationsanleitungen findet man kostenlos unter `http://www.mysql.com/`. Während der Entwicklung wurde MySQL in der Version 5.1 verwendet.
- **Apache Tomcat:** Tomcat wird als Serverseitige Laufzeitumgebung für das Projekt gebraucht und kann kostenlos über folgenden Link heruntergeladen werden: `http://tomcat.apache.org/` Entwickelt und getestet wurde dieses Projekt mit der Version 6.0.29.

Alle diese Komponenten sind für die gängigsten Betriebssystem- und Hardwareplattformen verfügbar, die Installation ist problemlos auf Apple Mac OS X, Microsoft Windows und Linux machbar. Es empfiehlt sich zudem ein Datenbank Administrationswerkzeug zu installieren. Dafür bietet sich beispielsweise das webbasierte Tool **phpMyAdmin** an, das frei verfügbar ist. Es kann über folgenden Link bezogen werden: `http://www.phpmyadmin.net/`

Auf die Installationsprozedur für diese drei Komponenten wird an dieser Stelle nicht weiter ins Detail gegangen. Für alle Komponenten sind auf der Webseite Installationsanleitungen für die jeweiligen Plattformen zu finden.

A.3. Installation und Konfiguration

Nach der Installation der oben beschriebenen Komponenten kann mit der Installation von Comitia begonnen werden. Als erster Schritt muss eine Datenbank und ein entsprechender User unter MySQL angelegt werden. Dies kann entweder in einem graphischen Administrationstool oder wie folgt auf der Kommandozeile geschehen:

A.3.1. Konfiguration der Datenbank

Listing A.2: Erstellen der Datenbank und eines Users

```
shell> mysql -uroot -p
Enter password: <rootpassword>
mysql> CREATE DATABASE <databasename>;
mysql> CREATE USER ‘<username>‘@‘localhost‘ IDENTIFIED BY ‘<password>‘;
mysql> GRANT ALL PRIVILEGES ON ‘<databasename>‘ . * TO ‘<username>‘@‘localhost‘;
```

Nun kann das initiale Schema in die Datenbank geladen werden. Dieses Schema liegt im Softwarepaket als Datei mit dem Namen comitia.sql vor und kann folgendermassen appliziert werden:

Listing A.3: Laden des Datenbankschemas in die Datenbank

```
shell> mysql -uroot -p <databasename> < comitia.sql
Enter password: <rootpassword>
```

Wie bei der Erstellung der Datenbank kann auch dieser Teil in einem Administrationswerkzeug der Wahl erledigt werden.

A.3.2. Konfiguration vom Tomcat

Da ein Teil der Sicherheitsmechanismen von Comitia auf X.509 Zertifikaten und SSL/TLS beruht, ist eine entsprechende Konfiguration von SSL/TLS unerlässlich. Diese Konfiguration geschieht im *server.xml* der Tomcat-Installation, üblicherweise im Verzeichnis *conf*. Darin findet sich typischerweise ein Block wie der folgende:

Listing A.4: Tomcat Original Konfiguration

```
<Connector port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           URIEncoding="UTF-8"
           redirectPort="8443" />
```

Hier muss nun noch ein HTTPS Connector hinzugefügt werden, indem zusätzlich einige SSL spezifische Angaben gemacht werden. Folgende Werte müssen konfiguriert und in der untenstehenden Beispielkonfiguration ersetzt werden:

- **Keystorepfad:** Pfad zum Keystore und Truststore
- **Keystorefile:** Keystore mit einem Serverzertifikat und zugehörigem Private Key. Für Testzwecke wurde nebst der Software ein entsprechendes Serverzertifikat mit dem Subjectname “localhost” mitgeliefert. Hier sollte ein offiziell für den Comitia-Server ausgestelltes Zertifikat referenziert werden.

- **Keystorepasswort:** Passwort für den Zugriff auf den Keystore. Für die mitgelieferten Keystores sind die Passwörter in der *README* Datei dokumentiert.
- **Keystoretyp:** Tomcat kann mit zwei Keystoretypen umgehen. Dies sind "PKCS12" und das Java spezifische "JKS"-Format. In der Regel liegen diese Keystores im PKCS12 Format vor.
- **Truststorefile:** In diesem Keystore sind die CA-Zertifikate eingetragen, die als vertrauenswürdig eingestuft werden. Wird der ganze Setup mit offiziell signierten Zertifikaten (sowohl Client- wie auch Serverzertifikate) durchgeführt, kann auch der mit Java mitgelieferte Truststore verwendet werden. Die gesamte Konfiguration des Truststore entfällt somit.
- **Truststorepasswort:** Passwort für den Zugriff auf den Truststore. Für die mitgelieferten Truststores sind die Passwörter in der *README* Datei dokumentiert.
- **Truststoretyp:** Wie beim Keystoretyp kann hier zwischen "JKS" und "PKCS12" gewählt werden. In der Regel wird für den Truststore "JKS" verwendet.

Nachfolgend findet sich eine Beispielkonfiguration. Die Werte in spitzen Klammern müssen gemäss der obigen Beschreibung angepasst werden.

Listing A.5: Beispielkonfiguration Tomcat

```
<Connector port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           URIEncoding="UTF-8"
           redirectPort="8443" />
  <Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
      maxThreads="150" scheme="https" secure="true"
           keystoreFile="<keystorepfad>/<keystorefile>"
           keystorePass="<keystorepasswort>" keystoreType="<keystoretyp>"
           truststoreFile="<keystorepfad>/<truststorefile>"
           truststorePass="<truststorepasswort>"
           truststoreType="<truststoretyp>"
           clientAuth="want" sslProtocol="TLS" />
```

Nach einem Neustart des Tomcats sollte der Webserver mit einem Browser sowohl über HTTP (Port 8080) wie auch über HTTPS (Port 8443) erreichbar sein. Wenn das Serverzertifikat nicht von einer offiziellen Certification Authority signiert worden ist, wird im Browser eine Sicherheitswarnung angezeigt, weil der Browser den Austeller des Serverzertifikats nicht kennt. Dies kann behoben werden, wenn entweder das Root CA Zertifikat der eigenen CA oder das Serverzertifikat selbst in den Truststore des Browsers geladen wird.

A.3.3. Deployment von Comitia

Nun kann die eigentliche Webapplikation im Tomcat installiert werden. Dazu muss die Datei *comitia.war* in das Verzeichnis *webapps* unterhalb des Basisverzeichnisses des Tomcat kopiert werden. Nach einem Neustart wird dieses Archiv entpackt und die Applikation automatisch in Betrieb genommen. Im Verzeichnis *webapps* wurde nun neben der Datei *comitia.war* ein Ordner *comitia* angelegt. Dieser enthält alle für den Betrieb der Applikation notwendigen Komponenten.

Nun muss noch die Konfiguration für die Datenbank in der Comitia-Applikation angepasst werden. Diese Konfiguration befindet sich im Tomcat Basisverzeichnis unter *webapps/comitia/WEB-INF/classes* in der Datei *comitiaNode.driver.xml*. In dieser Datei müssen die URL zur Datenbank, Benutzername und Passwort entsprechend der oben erstellten Datenbankkonfiguration angepasst werden.

Nach einem erneuten Neustart des Tomcats ist die Applikation serverseitig einsatzbereit.

B. Benutzerhandbuch

Um eine Abstimmung erfolgreich durchzuführen, muss eine bestimmte Reihenfolge eingehalten werden. Dieser Ablauf soll in diesem Kapitel im Detail erklärt werden. Der Ablauf entspricht dem in Kapitel 4 beschriebenen theoretischen Ablauf des CGS97 [1] Protokolls.

B.1. Vorbedingungen

Um in Comitia eine Abstimmung durchführen zu können, müssen grundsätzlich 3 Kriterien erfüllt sein:

- **Installation:** Das Comitia Softwarepaket muss, wie in der Installationsanleitung beschrieben, installiert und gestartet sein.
- **Zertifikate:** Jeder potentielle Wahladministrator, Wähler oder Trustee muss im Besitz eines gültigen Clientzertifikats sein, welches in seinem Browser installiert worden ist. Alle Zertifikate müssen gültig sein und durch den Webserver validiert werden können.
- **Personen sind bekannt:** In der Tabelle "person" in der Datenbank müssen alle potentiellen Teilnehmer einer Abstimmung mit den Merkmalen E-Mailadresse, Zertifikat und dem SHA-1 Fingerprint des Zertifikats hinterlegt worden sein.

B.2. Erfassen einer Abstimmung

Jeder erfasste Teilnehmer ist berechtigt, eine Abstimmung zu erstellen. Diese Funktion kann mit der Schaltfläche "Create Election" auf der Startseite der Applikation aufgerufen werden. Nach Betätigung dieser Schaltfläche wird der Wahladministrator in mehreren Schritten durch den Prozess für die Erstellung einer Abstimmung geleitet.

- **Fragestellung und Optionen:** Im ersten Schritt werden die Fragestellung und die Wahloptionen definiert. Die Wahloptionen werden im Textfeld "Available Options" eingeben und mit "Add" auf der Liste hinzugefügt. Jede Option kann mit "X" wieder von der Liste entfernt werden.

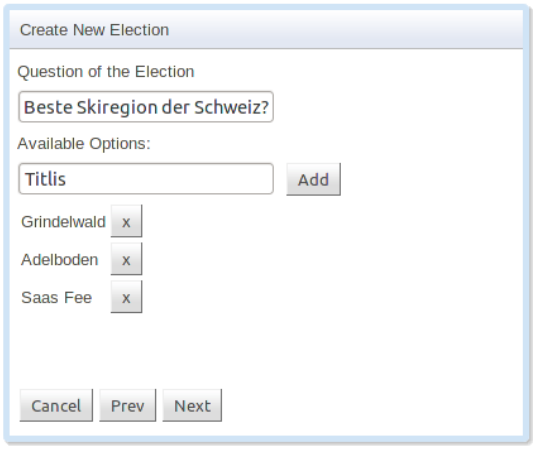

Abbildung B.1.: Fragestellung erfassen

- **Trustees:** In diesem Schritt kann der Wahladministrator die Trustees einer Abstimmung hinzufügen. Es gilt zu beachten, dass diese dem E-Voting-System bereits bekannt sein müssen, d.h. dass sie gemäss Installationsanleitung in der Datenbank aufgenommen wurden. Identifiziert werden die Trustees über die E-Mail Adresse. Wie beim Erfassen der Wahloptionen kann die E-Mail Adresse des Trustees im Textfeld erfasst und anschliessend mit einem Klick auf "Add" der Liste hinzugefügt werden.

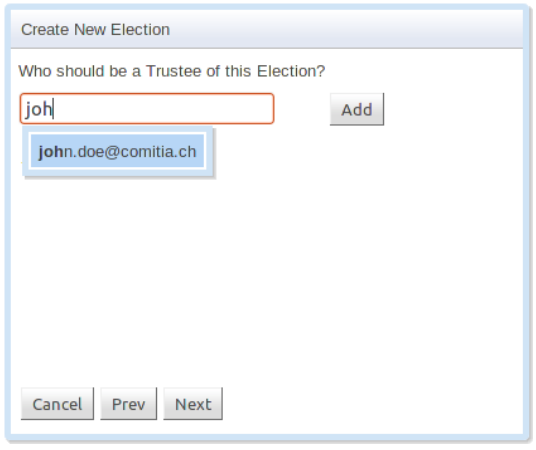

Abbildung B.2.: Trustees erfassen

- **Threshold:** Anschliessend muss ein Threshold definiert werden. Dieser Threshold definiert, wie viele korrekt agierende Trustees notwendig sind um das Endresultat entschlüsseln zu können. Dieser Wert darf nicht grösser sein als die Anzahl der definierten Trustees.

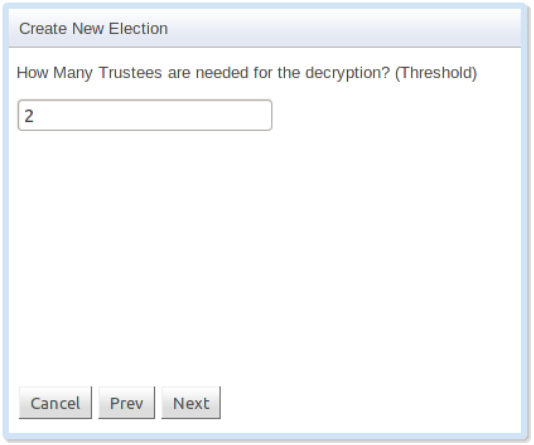

Abbildung B.3.: Threshold einstellen

- **Wähler:** Als letzter Schritt werden nun noch die Wähler der Abstimmung hinzugefügt. Die Erfassung dieser Wähler läuft analog dem Erfassen der Trustees ab.

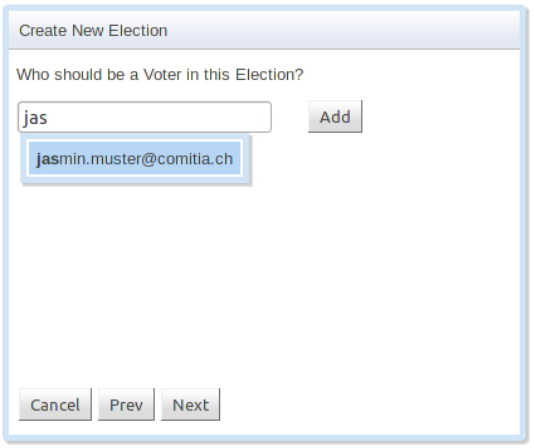

Abbildung B.4.: Wähler erfassen

Nachdem die Wähler erfasst wurden, werden die eingegebenen Daten mittels “Next” dem E-Voting-System gesendet und bestätigt.

B.3. Erstellen des Schlüsselmaterials durch die Trustees

Damit eine Abstimmung eröffnet werden kann, müssen die Trustees gemeinsam das nötige Schlüsselmaterial generieren. Dies geschieht mit Hilfe eines Java Rich Clients, welcher mit Betätigung der Schaltfläche “Participate” direkt von der Startseite aufgerufen werden kann. Diese Schaltfläche ist allerdings nur vorhanden, wenn einem angemeldeten Benutzer auch die Rolle als Trustee für eine Abstimmung zugewiesen wurde. Da

die Kommunikation auch bei dieser Komponente über HTTPS läuft, werden entsprechende Zertifikate vorausgesetzt. Beim Aufstarten des Trustee Clients öffnet sich zuerst ein Dialogfenster wo ein Keystore im PKCS12 Format und ein Truststore im JKS Format eingetragen werden muss. Diese Werte werden beim ersten Start des Trustee Clients persistent auf dem lokalen Dateisystem als *.comitita* im Homedirectory des Benutzers abgespeichert und müssen nur einmal eingegeben werden.

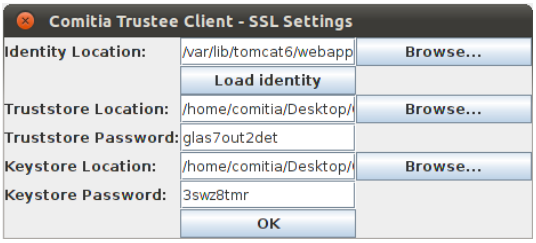

Abbildung B.5.: SSL Dialog

Nach dem Start des Clients erscheint die Oberfläche, die sich in 3 Teilen gliedert. Auf der linken Seite befinden sich Schaltflächen um bestimmte Aktionen auszulösen. Die rechte Hälfte ist in einen oberen und unteren Teil unterteilt. Oben erscheinen alle Abstimmungen, bei denen man sich als Trustee beteiligen kann. Im unteren Teil werden nach einem Klick auf “Get Trustees of Election” die zu der Abstimmung gehörenden Trustees mit dem jeweiligen Status dargestellt.

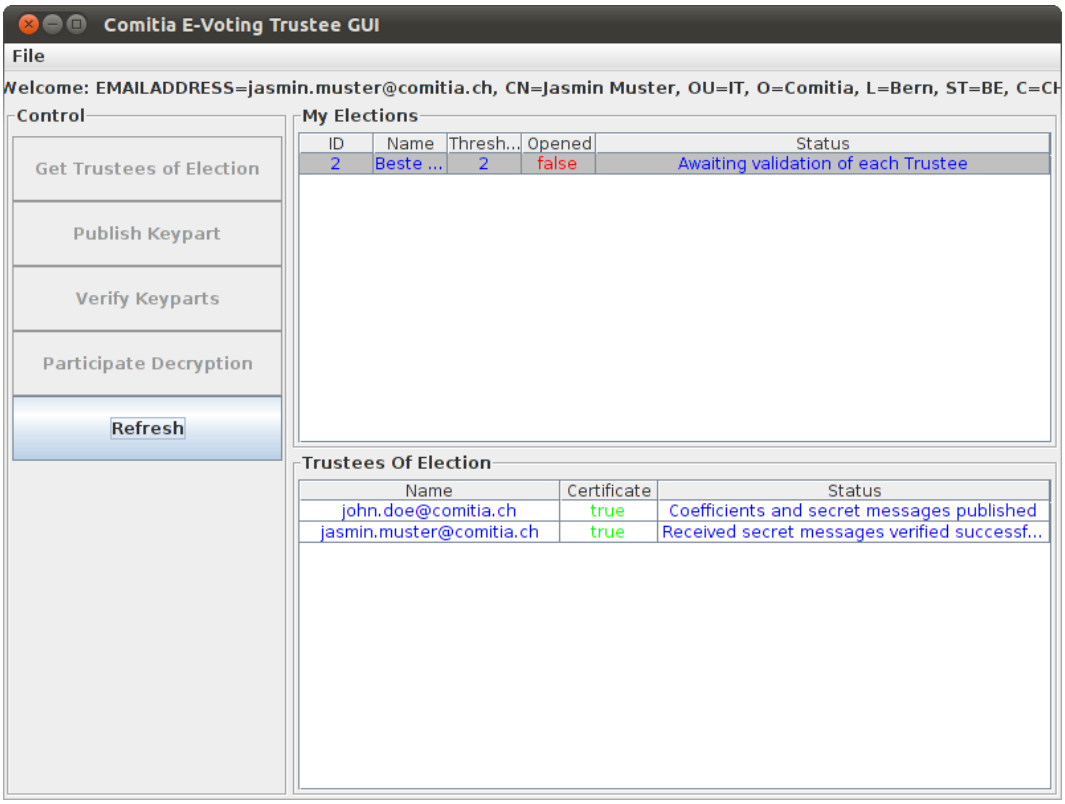

Abbildung B.6.: Comitia Trustee Client

Um das benötigte Schlüsselmaterial zu erstellen, muss jeder Trustee für die Abstimmung seinen Beitrag dazu leisten. Dies geschieht in zwei Schritten.

- **Schritt 1:** Jeder Trustee sendet seine Parameter als Beitrag für das gemeinsame Schlüsselpaar an das Comitia E-Voting-System indem er die Schaltfläche “Publish Keypart” betätigt.
- **Schritt 2:** Nachdem jeder Trustee Schritt 1 für eine Abstimmung durchgeführt hat, werden diese veröffentlichten Parameter durch alle Trustees gegenseitig mit “Verify Keyparts” geprüft.

B.4. Eröffnen der Abstimmung durch den Administrator

Nun ist alles bereit für das Eröffnen der Abstimmung. Der Administrator findet seine erstellte Abstimmung auf der Weboberfläche. Dort kann er mit einem Klick auf “Administer” einen Administrationsdialog öffnen, wo er die entsprechende Funktion findet um die Abstimmung zu eröffnen.

Abbildung B.7.: Eröffnen der Abstimmung

B.5. Abgabe von Stimmen durch die Wähler

Sobald die Abstimmung eröffnet wurde, kann jeder Wähler seine Stimme abgeben, indem er die Schaltfläche “Cast Ballot” für die gewünschte Abstimmung auf der Comitia Startseite betätigt. Es öffnet sich nun ein Dialog dessen Benutzerführung wie ein Assistent in mehrere Schritte unterteilt ist. Nach Auswahl einer Option wird der Wähler im nächsten Schritt dazu aufgefordert, seine Auswahl zu bestätigen. Tut er dies, wird im nächsten Schritt die Verschlüsselung der Stimme durchgeführt.

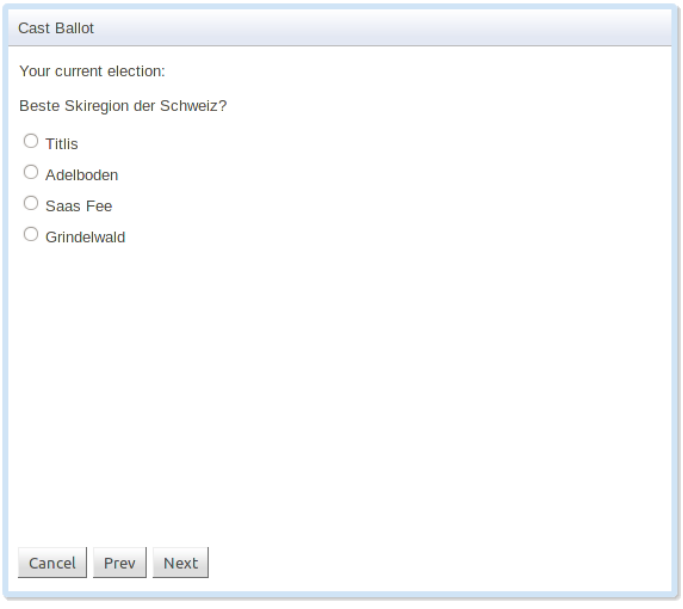

Abbildung B.8.: Option auswählen

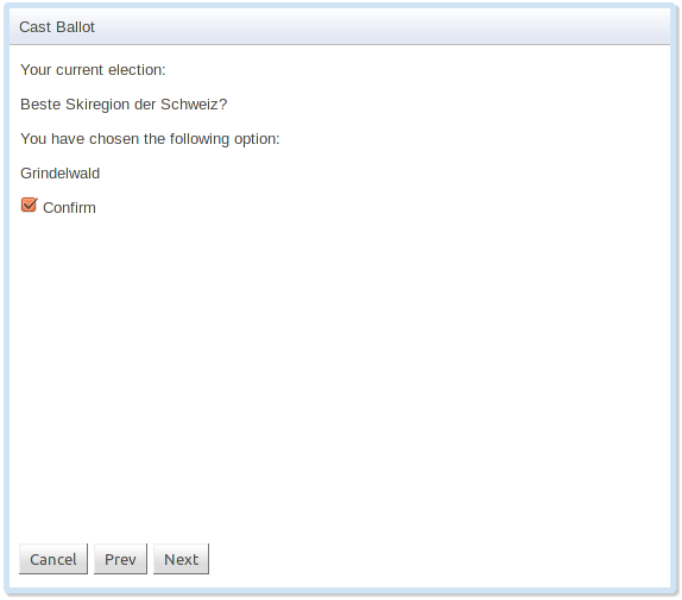

Abbildung B.9.: Option bestätigen

Da dieser Vorgang sehr rechenintensiv ist, kann dieser Vorgang einige Zeit in Anspruch nehmen. Folgende Faktoren beeinflussen die Dauer dieses Vorgangs:

- **Aktualität des Browsers:** Bis vor kurzem wurde Javascript im Webbrowser primär dazu eingesetzt, um einer Webseite mehr Funktionalität zu verleihen und nicht um rechenintensive Operationen durchzuführen. Dies ist sehr wahrscheinlich auch der Grund, weshalb die Performanceoptimierung der JavaScript-Komponente in Webrowsern nicht prioritär vorangetrieben wurde. Erst mit der Welle von neuartigen Rich Internet Applications haben die Browserhersteller einen starken Fokus auf die Verbesserung der JavaScript Performance gelegt. Ein Browser neueren Datums löst diese Aufgaben entsprechend schneller.
- **Anzahl Optionen der Abstimmung:** Da für jede Option quasi eine eigene Stimme erstellt wird, spielt dieser Faktor eine wesentliche Rolle. Die Verschlüsselung einer Stimme für eine Abstimmung mit 8 Optionen dauert fast doppelt so lange wie bei einer mit 4 Optionen.
- **Geschwindigkeit des Clientcomputers:** Natürlich spielt auch die Geschwindigkeit des Computers, mit dem der Wähler seine Stimme abgibt, eine Rolle. Maschinen neueren Datums meistern diese Aufgabe in kürzerer Zeit.

Dem Wähler wird die Wartezeit, bis seine Stimme verschlüsselt wurde, mit einem Statusbalken visualisiert.

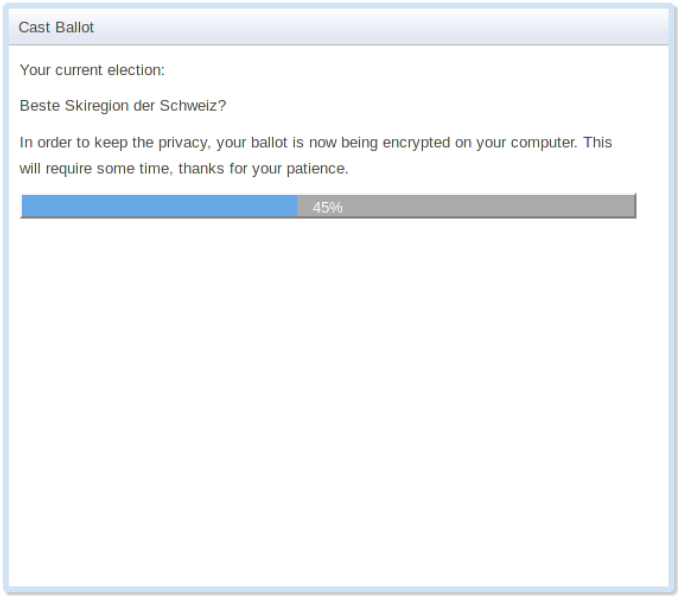

Abbildung B.10.: Verschlüsselung der Stimme

Nach dem Abschluss dieses Vorgangs ist die Stimme bereit zur Abgabe. Dem Wähler wird ein Fingerprint der Stimme (SHA-1 Hash) angezeigt. Dieser kann verwendet werden um sicherzustellen, dass die abgegebene Stimme nicht verändert wurde. Genau dieser Fingerprint sollte anschliessend auf dem Bulletin Board erscheinen. Ein Sachkundiger Wähler, der die Korrektheit des Ablaufs überprüfen will, hat zudem die Möglichkeit alle Daten, die dem E-Voting-System gesendet werden anzuzeigen und nachzuprüfen. Ist der Wähler zufrieden, kann er mittels Betätigung der Schaltfläche “Send” die Stimme dem E-Voting-System senden. Ab jetzt kommt die Serverseite des E-Voting-Systems ins Spiel, die vorhergehenden Schritte wurden ausschliesslich auf dem Client durchgeführt.

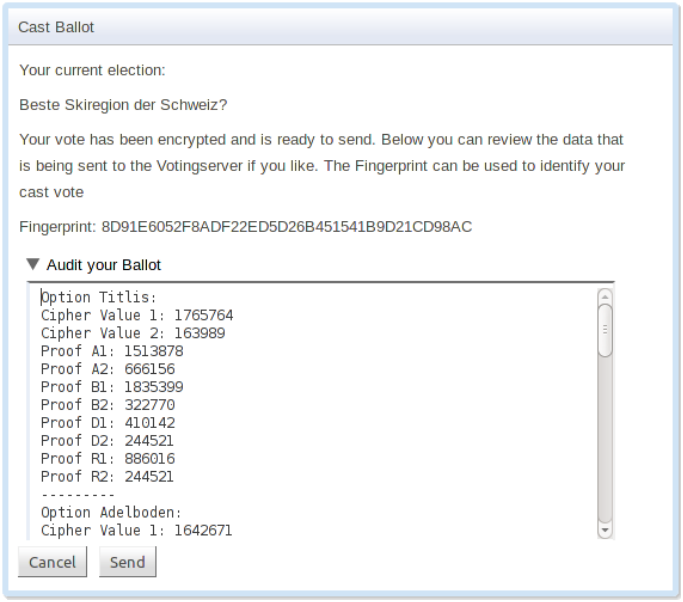

Abbildung B.11.: Auditierung des Ballots

B.6. Schliessen der Abstimmung und Tallying durch den Administrator

Sobald das Zeitfenster der Abstimmung abgelaufen ist, kann der Administrator die Abstimmung, analog zu der Eröffnung, wieder schliessen. Als erster Schritt der Auszählung muss nun der Administrator das sogenannte Tallying, das homomorphe Aufsummieren der Stimmen, auslösen. Ein Klick auf “Tally” erteilt dem E-Voting-System den Befehl, diese Summierung durchzuführen. Bei diesem Vorgang werden jegliche Stimmen überprüft und nur die gültigen in die Auszählung einbezogen. Diese Prüfung ist ziemlich rechenintensiv und kann, abhängig zur eingesetzten Schlüssellänge, etwas Zeit in Anspruch nehmen. Der erfolgreiche Abschluss wird mit einer entsprechenden Meldung bestätigt.

B.7. Verteiltes entschlüsseln durch die Trustees

Nachdem das Tallying durchgeführt worden ist, liegen die Resultate pro Option in verschlüsselter Form vor. Das Endresultat der Abstimmung kann nur entschlüsselt werden, wenn mindestens die Anzahl Trustees, die bei der Konfiguration als Threshold definiert wurden, korrekt zusammenarbeiten. Dazu muss jeder Trustee erneut den Trustee Client starten. Dort steht nun die Aktion “Participate Decryption” zu Verfügung. Beim Aufruf dieser Funktion wird eine Teilentschlüsselung an das E-Voting-System gesendet. Bei

jedem Eingang einer solchen Teilentschlüsselung prüft das E-Voting-System, ob bereits genug Trustees für die Entschlüsselung des Wahlresultats partizipiert haben. Sobald dies der Fall ist, kann das E-Voting-System die restlichen Operationen Durchführen, so dass das Resultat jeder Option im Klartext vorliegt.

B.8. Anzeigen des Resultats

Sobald das Resultat in Klartext verfügbar ist, kann es auf der Weboberfläche mittel drücken der Schaltfläche “Show Results” angezeigt werden. Im folgenden Fenster wird nun das Endresultat in Form eines Kuchendiagramms dargestellt.

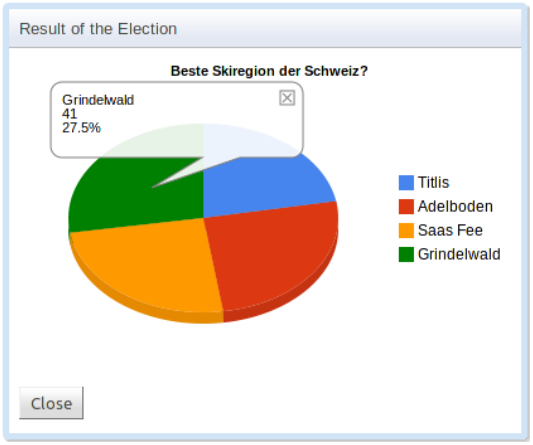

Abbildung B.12.: Anzeige des Resultats

B.9. Öffentliches Bulletinboard

Jeder, auch Personen die an der Abstimmung beteiligt waren, können den Ablauf der Wahl auf dem genannten “Bulletinboard” mitverfolgen und nachvollziehen. Dort werden sämtliche Werte, die öffentlich bekannt sind, publiziert. Zusätzlich sind Infos über Abstimmung, Trustees und Wähler ersichtlich. Mit Hilfe dieser Informationen kann jeder, nach CGS97 Protokoll [1] beschriebenen mathematischen Operationen, den korrekten Ablauf der Abstimmung verifizieren.

C. Pflichtenheft

Dieses Pflichtenheft beschreibt die Ziele, welche mit der angestrebten Lösung der Aufgabe der Bachelor Thesis zu erreichen sind, sowie die Anforderungen und Wünsche an die zu implementierende Applikation.

C.1. Ziele

C.1.1. Muss-Kriterien

- Implementierung eines funktionsfähigen E-Voting-Systems basierend auf dem Protokoll CGS97 mit folgenden Eigenschaften und Komponenten:
  - Einrichten einer Abstimmung durch den Administrator und die Trustees (Use Cases 1, 2, 3, 4)
  - Kontrolle des zeitlichen Ablaufs der Abstimmung durch den Wahladministrator (Use Cases 5, 8)
  - Abgabe der Stimme durch einen Wähler (Use Case 7)
  - Öffentlich zugängliches Brett (Bulletin Board), auf welchem der Verlauf der Abstimmung beobachtet werden kann. (Use Case 6)
  - Korrektes Auszählen der Stimmen und Veröffentlichung des Resultats (Use Cases 9, 10)
  - Abstimmung muss für jedermann überprüfbar sein (Use Cases 11, 12)
- Eine korrekte Implementation von CGS97 und dessen kryptographischen Bausteinen muss die Sicherheit des Systems gewährleisten

C.1.2. Soll-Kriterien

- Die Implementation legt den Fokus auf die Benutzerfreundlichkeit für den Wähler
- Eine Abstimmung soll einfach, intuitiv und effizient durchführbar sein

C.1.3. Kann-Kriterien

- Das E-Voting-System bietet dem Wähler die Möglichkeit, die Quittung seiner Stimmabgabe abzuspeichern (Use Case 14)
- Ein Wähler kann eine abgegebene Stimme korrigieren, solange die Abstimmung noch offen ist (Use Case 13)
- Bereitstellung eines Portals zur Einsicht aller erfassten Abstimmungen (Use Case 15)
- Die Dechiffrierung der abgegebenen Stimmen kann mittels geeignetem Algorithmus optimiert werden, z.B. Paillier oder Babystep-Giantstep-Algorithmus
- Graphisches Userinterface für die Administration einer Abstimmung (Erstellung, Verwalten der Wähler und Trustees, Abschliessen der Abstimmung, Stimmenauszählung)

C.1.4. Abgrenzungskriterien

- Die Möglicheit, dass ein Wähler aus den angebotenen Optionen mehrere Optionen auswählen kann, wird mit diesem E-Voting-System nicht realisiert ($\mathbb{M}$ aus $\mathbb{N}$)
- Es ist nicht vorgesehen, die Administration einer Abstimmung (Erstellung, Verwalten der Wähler und Trustees, Abschliessen der Abstimmung, Stimmenauszählung) über ein graphisches Userinterface zu ermöglichen. Diese Aufgabe bleibt einem technisch versierten Benutzer vorenthalten, da die Verwaltung mittels Konfigurationsdateien ermöglicht wird
- Es werden keine sicherheitstechnischen Massnahmen umgesetzt, die nicht ausdrücklich vom CGS97 Protokoll verlangt werden (z.B. Abwehr von DoS-Angriffen, Replikation oder manipulation von Datenbeständen)
- Die entwickelte Software ist nicht für den produktiven Einsatz konzipiert, sondern als Prototyp resp. Fallstudie/Machbarkeitsstudie gedacht
- Da das zugrunde liegende Protokoll CGS97 nicht quittungsfrei ist, wird auch das System nicht erpressungsfrei sein
- Es ist nicht vorgesehen, dass Wähler nachträglich in einer Abstimmung aufgenommen werden können, wenn diese bereits freigeschaltet worden ist.

C.2. Produkteinsatz

C.2.1. Anwendungsbereiche

Die entwickelte Software ist nicht für den produktiven Einsatz konzipiert, sondern als Prototyp resp. Fallstudie/Machbarkeitsstudie gedacht. Der Anwendungsbereich beschränkt sich dadurch auf die Demonstration eines E-Voting-Systems, welchem das Protokoll von CGS97 zugrunde liegt.

C.2.2. Zielgruppen

Aus rein applikatorischer Sicht unterscheiden wir drei Zielgruppen. Dies sind einerseits die Wähler, also die Endbenutzer der Applikation, die Administratoren und die Trustees, welche an der Schlüsselgenerierung und der Entschlüsselung der Stimmen beteiligt sind.

Da die Benutzerfreundlichkeit des E-Voting-Systems vor allem für den Wähler optimiert wird, nimmt diese Zielgruppe die höchste Priorität in Anspruch. Diese Zielgruppe muss über keine Kenntnisse zu Kryptographie und im speziellen zum Protokoll von CGS97 verfügen, um die Applikation bedienen zu können, d.h. an einer Abstimmung teilnehmen zu können.

Die Administration einer Abstimmung bleibt technisch versierten Personen vorbehalten, die zudem ein Grundverständnis des Protokolls von CGS97 aufweisen müssen. Dasselbe gilt für die Trustees.

Da das Endprodukt dieses Projekts als Prototypimplementation für die E-Voting Forschungsgruppe der Berner Fachhochschule dient, können die Forschungsgruppenmitglieder auch als Zielgruppe angesehen werden. Es ist vorgesehen, diese Implementation in weiteren Projekten zu verbessern.

C.3. Produktübersicht

Als Endprodukt soll eine Applikation entstehen, über die eine elektronische Abstimmung gemäss dem CGS97 Protokoll erstellt, durchgeführt und ausgezählt werden kann. Für einen Wähler soll es ohne Kenntnisse von Kryptographie möglich sein, erfolgreich an einer Abstimmung teilnehmen zu können.

C.4. Produktfunktionen

In diesem Kapitel werden die einzelnen Funktionen eines solchen Systems beschrieben.

C.4.1. Abstimmung einrichten

- Erstellen einer Abstimmung
- Fragestellung definieren (z.B. Wo gehen wir essen? Wer wird Gemeindepräsident?)
- Mögliche Antworten definieren (bei der Abstimmung kann nur eine dieser Antworten gewählt werden)
- Erfassen von Wählern und Trustees
- Trustees führen Schlüsselgenerierung durch
- Möglichkeit eine Abstimmung zu eröffnen/schliessen
- Vor Eröffnung und nach Schliessung ist keine Stimmabgabe möglich

C.4.2. Durchführung einer Abstimmung

- Wähler wird zu einer Abstimmung eingeladen
- Nach Eröffnung und bis zur Schliessung der Wahl kann der Voter seine Stimme abgeben
- Der Wähler identifiziert sich (kann auch nach Erstellung der Stimme geschehen)
- Der Wähler kann eine der angebotenen Optionen auswählen
- Die Stimme wird gemäss CGS97 verschlüsselt und mit einem Beweis versehen
- Die Stimme wird an das E-Voting-System gesendet
- Der Wähler kann alle abgegebenen Stimmen auf einem öffentlichen zugänglichen Brett (Bulletin Board) einsehen

C.4.3. Auszählen der Stimmen und Veröffentlichung des Resultats

- Die verschlüsselten Stimmen werden summiert
- Die Trustees entschlüsseln gemeinsam das Abstimmungsresultat
- Das Abstimmungsresultat wird auf dem öffentlichen Board publiziert

C.5. Produktdaten

Benutzer- und Abstimmungsrelevante Daten werden persistent in einer Datenbank gespeichert.

Die manuelle Veränderung von Daten über ein geeignetes Datenbank-Management-System wird explizit erlaubt, da es sich bei unserer Implementation um eine Machbarkarkeitstudie handelt. Es kann für Test-Szenarien durchaus sinnvoll sein, dass gewisse Daten gelöscht, verändert oder manuell eingefügt werden können. Als Beispiels sei das Beobachten der Veränderung des Abstimmungsresultats zu nennen, wenn eine Stimme gelöscht wird. In diesem Fall wäre es nur hinderlich, müsste das ganze Szenario noch einmal von Anfang an durchgespielt werden.

Entsprechende Funktionen für die Datenmanipulation in der Applikation oder über das Userinterface sind nicht vorgesehen.

C.6. Qualitätsanforderungen

- Einhaltung der Sicherheitsmerkmale entsprechend den Vorgaben von CGS97
- Versionierung der Dokumentation und des Quellcodes mittels Subversion
- Einsatz von etablierten und aktuellen Entwicklungsumgebungen sowie Frameworks
- Es wird ein “Test Driven Development” angestrebt, um die korrekte Funktionsweise des Programms so weit wie möglich zu gewährleisten

C.7. Benutzungsoberfläche

- Die Benutzeroberfläche für den Wähler ist webbasiert, sie soll intuitiv und benutzerfreundlich sein

C.8. Authentisierung und Authorisierung

- Um einen Benutzer zu identifizieren wird ein gängiger Authentisierungsmechanismus wie Username/Passwort oder digitale Zertifikate verwendet
- Aufgrund der Authentisierung des Users werden entsprechende Berechtigungen erteilt

C.9. Technische Produktumgebung

C.9.1. Systemdesign

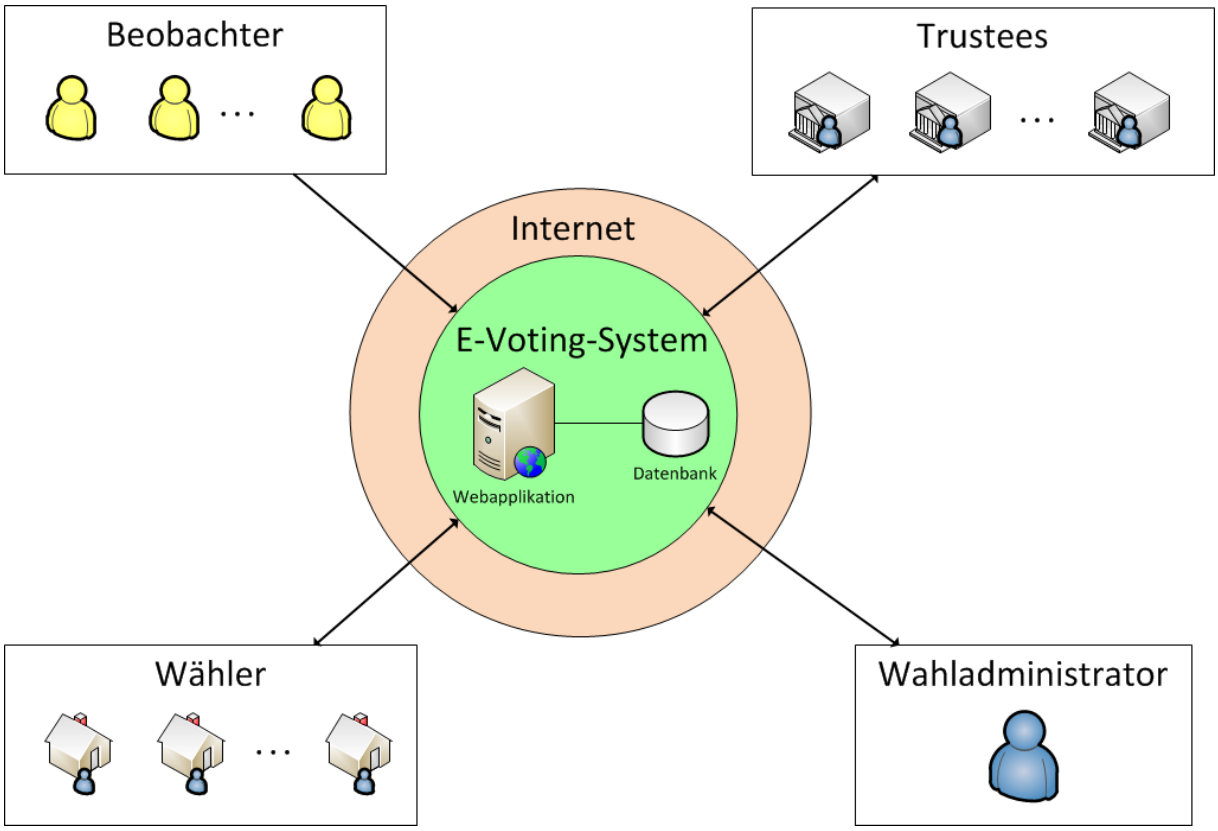

Abbildung C.1.: Systemdesign

C.9.2. Software

Client

Voraussetzung für die Stimmabgabe ist ein aktueller Browser und eine Internetverbindung. Der Browser muss unabhängig des Betriebssystems folgende Bedingungen erfüllen:

- aktueller Browser (Stand Ende 2010)
- JavaScript muss im Browser aktiviert sein
- Cookies müssen im Browser aktiviert sein

Da die Trustees ziemlich rechenintensive Aufgaben zu erfüllen haben wird auf dem Client der Trustees ein Java Runtime Environment mindestens in der Version 1.6 vor-

ausgesetzt. Es wird in Betracht gezogen, die Softwarekomponente für die Trustees als Java Rich Client Anwendung zu implementieren. Ein JavaScript, das im Browser läuft, ist für diese Aufgaben aufgrund der Annahme, dass die Dauer für Berechnungen zu lange sein wird, nicht geeignet.

Server

Die Applikation wird serverseitig in Java implementiert. Folgende Komponenten werden benötigt, um die Applikation auf einer Serverumgebung zu betreiben:

- Java Runtime Environment (mindestens Version 1.6)
- Apache Tomcat Servlet Container (mindestens Version 6.0)
- MySQL Datenbankserver (mindestens Version 5.1)

Grundsätzlich sind Servlet Container wie auch die Datenbank austauschbar, jedoch wird die Applikation mit diesen Softwarekomponenten entwickelt und getestet. Alle eingesetzten Komponenten laufen auf verschiedenen Betriebssystemen, die Wahl des Betriebssystems wird somit auf die von den Komponenten unterstützten Betriebssysteme beschränkt.

C.9.3. Hardware

Client

Grundsätzlich muss auf dem Client ein Browser in einer oben genannten Ausprägung lauffähig sein. Somit sind die Anforderungen an die Hardware gleich der Hardwareanforderungen des auf dem Client eingesetzten Browsers.

Server

Die Serverinfrastruktur muss in der Lage sein, die oben genannte Serversoftware (Apache Tomcat und MySQL Datenbankserver) mit einer angemessenen Performance ausführen zu können. Grundsätzlich ist es auch möglich, Servlet Container und Datenbankserver physikalisch auf mehrere Server zu verteilen um Sicherheit oder Performance der Applikationen zu erhöhen. Dies ist aber nicht Bestandteil dieser Arbeit.

C.9.4. Organisatorische Rahmenbedingungen

Es ist angedacht, die Authentisierung der Benutzer mit einem digitalen Zertifikat zu bewerkstelligen. Deshalb wird vorausgesetzt, dass der Wähler und jeder Trustee über

ein gültiges X.509 Zertifikat (z.B. SuisseID) verfügt, welches vom Server als vertrauenswürdig eingestuft wird. Dieses Zertifikat wird verwendet um die Identität der beteiligten Parteien festzustellen und entsprechende Zugriffsrechte zu erteilen.

C.9.5. Produktschnittstellen

Die Implementation dieses E-Voting-Systems wird als Produkt einer in sich abgeschlossenen Anwendung ohne Schnittstellen zu anderen Services sein.

C.10. Spezielle Anforderungen an die Entwicklungsumgebung

Allgemein sollen die Anforderungen unabhängig des Betriebssystems sein. So verwenden die Bachelor-Studenten auch unterschiedliche Betriebssysteme (u.a. Windows 7 und Mac OS X).

C.10.1. Software

Die Applikation wird mit Java realisiert.

- Eclipse (Version 3.5 oder höher) als IDE
- Google Web Toolkit (GWT) (Version 2.0.4) als Web-Framework
- Apache Cayenne für Objektrelationale Abbildung (ORM)
- log4j als Framework für das Logging
- MySQL-Datenbankserver
- Subversion-Client für den Zugriff auf den Versionierungsserver
- LaTeX-Umgebung für das Verfassen der schriftlichen Bachelor-Thesis

C.10.2. Hardware

Die Hardware für die Entwicklungsumgebung wird nicht genau vorgegeben. Auf heute verfügbarer Hardware ist oben genannte Software problemlos einsatzfähig.

C.10.3. Orgware

Für das Projektmanagement und als Kommunikationsplattform wird “Origo” der ETH Zürich eingesetzt (siehe http://origo.ethz.ch). Die verwendeten Module sind Blog, Wiki und Versionierungsserver (Subversion). Zu einem späteren Zeitpunkt der Umsetzung

ist der Einsatz des Issue-Tracking über “Origo” denkbar. Die Kommunikation zwischen Betreuer und Studenten findet zum Teil auch per E-Mail statt.

“Google Wave” wird im Gegensatz zum “Projekt 2” nicht mehr verwendet, da Google per Ende 2010 diesen Dienst einstellen wird.

C.10.4. Entwicklungsschnittstellen

Die Schnittschnellen zwischen Entwicklungsumgebung und externen Systeme sind

- Zugriff auf Versionierungsserver (Schnittstelle zu Subversion)
- Zugriff zum Issue Tracking (Schnittstelle über Mylyn)

D. Use Cases

In diesem Kapitel werden die Interaktionen zwischen Anwendern (Akteure) und dem E-Voting-System anhand von Use Cases beschrieben. Folgende Akteure sieht das System vor:

- Wähler: Eine Person die sich an einer Abstimmung beteiligt.
- Wahladministrator: Ist für die Erstellung und den Ablauf der Abstimmung verantwortlich. Der Wahladministrator kann auch als Wähler an der Abstimmung teilnehmen.
- Trustee: Vertrauenswürdige Drittpartei, von der es eine in der Abstimmung definierte Anzahl braucht, um das Wahlresultat zu entschlüsseln (Threshold-Krypto-Verfahren). Ein Trustee kann auch als Wähler an der Abstimmung teilnehmen.
- Beobachter: Person, die sich nicht aktiv an der Abstimmung beteiligt, jedoch Zugriff auf das Bulletin Board hat und den Wahlablauf verfolgen und überprüfen kann.

Use Case Diagramm

Das folgende Use Case Diagramm visualisiert die Interaktionen zwischen den einzelnen Akteuren und dem E-Voting-System.

Abbildung D.1.: Use Case Diagramm

D.1. Use Case 1 - Abstimmung erfassen

Referenzierte Use Cases

Keine

Beschreibung

Erfassung der Daten für die Abstimmung.

Akteure

- Wahladministrator

Vorbedingungen

Die Abstimmung wurde eröffnet.

Ergebnisse und Nachbedingungen

Die Abstimmungsfrage, Beschreibung und Wahloptionen sind erfasst.

Ablauf

Der Wahladministrator

- eröffnet eine Abstimmung
- erfasst Fragestellung und Beschreibung
- erfasst zwei oder mehr Wahloptionen

Besondere Anforderungen

Das System muss verfügbar sein.

Häufigkeit

Jedes mal, wenn eine neue Abstimmung eröffnet wird.

D.2. Use Case 2 - Wähler erfassen

Referenzierte Use Cases

Keine

Beschreibung

Der Wahladministrator definiert die Wähler und ordnet sie der Abstimmung zu.

Akteure

- Wahladministrator

Vorbedingungen

Die Abstimmungsfrage, Beschreibung und Wahloptionen wurden erfolgreich erfasst.

Ergebnisse und Nachbedingungen

Die Wähler sind erfasst.

Ablauf

- Wahladministrator definiert die zur Abstimmung zugelassenen Wähler
- Wahladministrator ordnet die Wähler der Abstimmung zu

Besondere Anforderungen

Das System muss vefügbar sein.

Häufigkeit

Jedes mal, wenn eine Abstimmung eröffnet wird.

D.3. Use Case 3 - Trustees erfassen

Referenzierte Use Cases

Keine

Beschreibung

Der Wahladministrator definiert die Trustees und ordnet sie der Abstimmung zu. Die Trustees werden aufgefordert, ihre Schlüsselpaare zu generieren und zusammen den öffentlichen Schlüssel zu berechnen.

Akteure

- Wahladministrator
- Trustees

Vorbedingungen

Die Wähler wurden erfasst.

Ergebnisse und Nachbedingungen

Die Trustees sind erfasst.

Ablauf

- Wahladministrator definiert die Trustees
- Wahladministrator ordnet die Trustees der Abstimmung zu
- Wahladministrator definiert die Anzahl der Trustees, welche für eine Entschlüsselung des Resultats gebraucht werden

Besondere Anforderungen

Das System muss verfügbar sein.

Häufigkeit

Einmal pro Abstimmung

D.4. Use Case 4 - verteilte Schlüsselgenerierung

Referenzierte Use Cases

- Use Case 3 - Trustees erfassen

Beschreibung

Generierung eines Schlüsselpaars, wobei der private Schlüssel unter den Trustees aufgeteilt wird.

Akteure

- Trustees

Vorbedingungen

Die Trustees wurden erfasst.

Ergebnisse und Nachbedingungen

- Jeder Trustee generiert und seinen Teilschlüssel
- Die Erfassung der Absimmung ist abgeschlossen

Ablauf

- Jeder Trustee generiert ein Teil-Schlüssel für sich und jeden anderen Trustee
- Die zur Überprüfung verwendeten Werte des Teil-Schlüssel werden veröffentlicht, ohne Rückschlüsse auf das Secret zuzulassen
- Jeder Trustee sendet jedem anderen Trustee den jeweiligen Teil des Teil-Schlüssel über einen sicheren Kanal
- Die Trustees überprüfen die erhaltenen Teil-Schlüssels mit Hilfe der zuvor publizierten Werte
- Jeder Trustee kann nun den öffentlichen Schlüssel berechnen

Besondere Anforderungen

Dieser Ablauf erfordert zwei Interaktionen von allen Trustees mit dem E-Votingsystem.

Häufigkeit

Einmal pro Abstimmung

D.5. Use Case 5 - Abstimmung freischalten

Referenzierte Use Cases

- Use Case 1 - Abstimmung erfassen
- Use Case 2 - Wähler erfassen
- Use Case 3 - Trustees erfassen

Beschreibung

Die Abstimmung wird eröffnet und damit den Wählern die Stimmabgabe ermöglicht.

Akteure

- Wahladministrator

Vorbedingungen

Die Abstimmung wurde erfasst.

Ergebnisse und Nachbedingungen

Die Abstimmung ist freigeschaltet.

Ablauf

- Wahladministrator schaltet die Abstimmung frei

Besondere Anforderungen

Das System muss verfügbar sein.

Häufigkeit

Jedes Mal, wenn eine Abstimmung eröffnet wird.

D.6. Use Case 6 - Bulletin Board einsehen

Referenzierte Use Cases

- Use Case 5 - Abstimmung freischalten

Beschreibung

Nach Eröffnung einer Abstimmung ist ein öffentlich zugängliches Bulletin Board verfügbar. Darauf werden alle eingegangenen Stimmen und die Interaktionen zwischen den Trustees publiziert, um den korrekten Ablauf einer Abstimmung zu protokollieren.

Akteure

- Wähler
- Beobachter

Vorbedingungen

Die Abstimmung wurde freigeschaltet.

Ergebnisse und Nachbedingungen

keine

Ablauf

- Ein Wähler oder ein Beobachter ruft die Informationen auf dem Bulletin Board ab

Besondere Anforderungen

Das System muss verfügbar sein.

Häufigkeit

Sehr häufig

D.7. Use Case 7 - Stimme abgeben

Referenzierte Use Cases

- Use Case 5 - Abstimmung freischalten

Beschreibung

Ein Wähler nimmt an einer Abstimmung teil. Dazu wählt er eine Option aus, verschlüsselt diese, legt einen Zero-Knowledge-Beweis gemäss dem CGS97 Protokoll bei und sendet diese Daten an das E-Voting-System.

Akteure

- Wähler

Vorbedingungen

Die Abstimmung wurde freigeschaltet.

Ergebnisse und Nachbedingungen

- Die Stimme wird zusammen mit dem Zero-Knowledge-Beweis auf dem E-Voting-System gespeichert und ist entsprechend auf dem Bulletin Board sichtbar

Ablauf

- Der Wähler identifiziert sich am E-Voting-System
- Die offenen Abstimmungen werden aufgelistet

- Auswahl einer Abstimmung
- Die Fragestellung und die möglichen Optionen werden angezeigt
- Der Wähler trifft eine Auswahl
- Der Wähler verschlüsselt die Stimme und hängt einen Zero-Knowledge-Beweis an
- Die verschlüsselte Stimme und der Zero-Knowledge-Beweis werden an das E-Voting-System gesendet
- Die Stimme wird im E-Voting-System abgelegt und ist auf dem Bulletin Board publiziert

Besondere Anforderungen

Das System muss verfügbar sein.

Häufigkeit

Sehr häufig

D.8. Use Case 8 - Abstimmung schliessen

Referenzierte Use Cases

- Use Case 5 - Abstimmung freischalten

Beschreibung

Der Wahladministrator schliesst eine Abstimmung.

Akteure

- Wahladministrator

Vorbedingungen

Die Abstimmung wurde freigeschaltet.

Ergebnisse und Nachbedingungen

Die Abstimmung ist geschlossen.

Ablauf

- Der Wahladministrator meldet sich am System an
- Der Wahladministrator schliesst die Wahl

Besondere Anforderungen

Das System muss verfügbar sein.

Häufigkeit

Einmal pro Abstimmung

D.9. Use Case 9 - Stimmen summieren

Referenzierte Use Cases

- Use Case 8 - Abstimmung schliessen

Beschreibung

Die verschlüsselten Stimmen werden auf ihre Gültigkeit überprüft und homomorph aufsummiert. Als Ergebnis erhält man das verschlüsselte Resultat der Abstimmung.

Akteure

- Trustees

Vorbedingungen

Die Abstimmung wurde geschlossen.

Ergebnisse und Nachbedingungen

Die Stimmen sind summiert.

Ablauf

- Die gültigen und verschlüsselten Stimmen werden homomorph summiert

Besondere Anforderungen

Das System muss verfügbar sein.

Häufigkeit

Einmal pro Abstimmung

D.10. Use Case 10 - Abstimmungsresultat entschlüsseln und publizieren

Referenzierte Use Cases

- Use Case 9 - Stimmen summieren

Beschreibung

Die Trustees entschlüsseln gemeinsam das Abstimmungsresulat, das bis dahin nur in verschlüsselter Form vorhanden ist (Use Case 9 - Stimmen summieren).

Akteure

- Trustees

Vorbedingungen

Die Stimmen wurden summiert.

Ergebnisse und Nachbedingungen

Das Abstimmungsresultat ist publiziert.

Ablauf

- Jeder Trustee beweist mit einem Zero-Knowledge-Beweis, dass er den korrekten Teil-Schlüssel für die Entschlüsselung der Abstimmung verwendet
- Jeder Trustees führt eine Teilentschlüsselung durch und kann dann mit allen Teilentschlüsselungen das Abstimmungsergebnis vollständig herstellen

Besondere Anforderungen

- Dieser Schritt erfordert eine genügende Anzahl von korrekt nach Protokoll agierenden Trustees

Häufigkeit

Einmal pro Abstimmung

D.11. Use Case 11 - Abstimmungsresultat überprüfen

Referenzierte Use Cases

- Use Case 10 - Abstimmungsresultat entschlüsseln und publizieren

Beschreibung

Ein Wähler oder ein Beobachter kann den korrekten Ablauf der Abstimmung überprüfen, indem er die publizierten Daten auf dem Bulletin Board ausliest und verifiziert.

Akteure

- Wähler
- Beobachter

Vorbedingungen

Das Abstimmungsresultat wurde publiziert.

Ergebnisse und Nachbedingungen

Das Abstimmungsresultat ist überprüft.

Ablauf

- Wähler oder Beobachter holt Daten vom Bulletin Board
- Wähler oder Beobachter verifiziert Ablauf der Wahl gemäss dem CGS97 Protokoll

Besondere Anforderungen

Daten sind auf dem Bulletin Board verfügbar

Häufigkeit

Nicht vorhersehbar

D.12. Use Case 12 - Abstimmungsresultat einsehen

Referenzierte Use Cases

- Use Case 10 - Abstimmungsresultat entschlüsseln und publizieren

Beschreibung

Ein Wähler oder ein Beobachter nimmt Einsicht in das Resultat der Abstimmung.

Akteure

- Wähler
- Beobachter

Vorbedingungen

Das Abstimmungsresultat wurde publiziert

Ergebnisse und Nachbedingungen

keine

Ablauf

- Ein Wähler oder ein Beobachter Informiert sich auf dem Bulletin Board wie die Abstimmung ausgegangen ist

Besondere Anforderungen

Das System muss verfügbar sein.

Häufigkeit

Nicht vorhersehbar

D.13. Use Case 13 - Stimmabgabe korrigieren

Referenzierte Use Cases

- Use Case 7 - Stimme abgeben
- Use Case 8 - Abstimmung schliessen

Beschreibung

Ein Wähler möchte nach der Stimmabgabe die ausgewählte Option korrigieren. Er hat nun die Möglichkeit die Abstimmung zu wiederholen und die Stimme erneut zu senden. Das E-Voting-System wird nun die bestehende Stimme markieren und für ungültig erklären. Dies kann jedoch nur geschehen, wenn die Abstimmung noch nicht geschlossen wurde (Use Case 8)

Akteure

- Wähler

Vorbedingungen

Die Abstimmung wurde freigeschaltet und es wurde bereits eine Stimme abgegeben.

Ergebnisse und Nachbedingungen

Der Wähler hat eine neue Stimme erstellt und an das E-Voting-System gesendet.

Ablauf

- Ablauf gemäss Use Case 7
- Die alte Stimme wird als ungültig markiert und somit bei der Auszählung nicht berücksichtigt

Besondere Anforderungen

Das System muss verfügbar sein.

Häufigkeit

Nicht vorhersehbar

D.14. Use Case 14 - Stimmencode abspeichern

Referenzierte Use Cases

- Use Case 7 - Stimme abgeben

Beschreibung

Um die abgegebene Stimme identifizieren zu können, wird bei der Stimmabgabe ein eindeutiger Stimmencode generiert. Der Wähler muss diesen Stimmencode ohne Einwirkung des E-Voting-Systems speichern können.

Akteure

- Wähler

Vorbedingungen

Die Abstimmung wurde freigeschaltet und es wurde bereits eine Stimme abgegeben.

Ergebnisse und Nachbedingungen

Der Wähler hat den Stimmencode abgespeichert.

Ablauf

- Der Wähler nimmt an der Abstimmung teil (Use Case 7)
- Der dabei entstandene Stimmencode wird ohne Einwirkung des E-Voting-Systems beim Wähler gespeichert.

Besondere Anforderungen

Das System muss verfügbar sein.

Häufigkeit

Nicht vorhersehbar

D.15. Use Case 15 - Abstimmungsportal einsehen

Referenzierte Use Cases

- Use Case 1 - Abstimmung erfassen

Beschreibung

Nachdem eine Abstimmung erfasst wurde, ist sie für jeden auf einem Portal ersichtlich. Dies ist auch der Fall, wenn die Abstimmung noch nicht eröffnet wurde.

Akteure

- Wähler
- Beobachter

Vorbedingungen

keine

Ergebnisse und Nachbedingungen

keine

Ablauf

- Ein Wähler oder ein Beobachter informiert sich auf dem Abstimmungsportal über die Abstimmung

Besondere Anforderungen

Das System muss verfügbar sein.

Häufigkeit

Nicht vorhersehbar

E. Bezeichnungen

Die Definition der Bezeichnungen, die in dieser Arbeit verwendet werden, befinden sich im jeweiligen Kapitel bzw. Abschnitt. Zur besseren Übersicht sind hier die wichtigsten Bezeichnungen aufgelistet.

Bezeichnung	Bedeutung
m	Eine unverschlüsselte Nachricht (Plaintext)
$c(m)$	Die verschlüsselte Nachricht m (Cipher)
p	Domänenparameter des ElGamal-Kryptosystem
q	Domänenparameter des ElGamal-Kryptosystem
g	Generator im ElGamal-Kryptosystem
k	Zufallswert im ElGamal-Kryptosystem
$s \in \mathbb{Z}_q$	Der private oder geheime Schlüssel (Private Key oder Secret Key) im ElGamal-Kryptosystem
h	Der öffentliche Schlüssel (Public Key) im ElGamal-Kryptosystem $h = g^k \pmod p$
(x, y)	Ein ElGamal-Cipher mit $x = g^k \pmod p$ $y = h^k \cdot m \pmod p$
t	Threshold im E-Voting-System
A_j	Der Trustee j
$f_j()$	Die Polynomialfunktion vom Grad $t-1$ des Trustees A_j $f_j() = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \ldots + a_{t-1} \cdot x^{t-1}$
a_i	Die Koeffizienten einer Polynomialfunktion
$f_j(i)$	"Share" für den Trustee A_i berechnet mit $f_j()$ des Trustees A_j
s_j	Der private Teilschlüssel (Private Key Part) des Trustees j $s_j = \sum_{i=1}^{n} f_j(i)$
h_j	Der öffentlicher Teilschlüssel (Public Key-Part) des Trustees j $h_j = g^{s_j} \pmod p$
λ_j	Der Lagrange-Koeffizient des Trustees A_j
w_j	Durch den Trustee A_j berechnete Teil-Entschlüsselung

Glossar

Apache Tomcat Freie Server-Software für die Ausführung von Java-Webapplikationen (http://tomcat.apache.org/). 89

Babystep-Giantstep-Algorithmus Algorithmus zur Berechnung des Diskreten Logarithmus in einer zyklischen Gruppe (http://de.wikipedia.org/wiki/Babystep-Giantstep-Algorithmus). 84

Bulletin Board Ein öffentlich zugängliches Brett, auf welchem die abgegebenen verschlüsselten Stimmen eingesehen werden können. 83, 86, 92, 98–100, 103, 104

Certification Authority Eine Organisation, die digitale Zertifikate herausgibt. 21, 65, 71

CGS97 Siehe auch Abschnitt E-Voting mit homomorpher Verschlüsselung (CGS97). CGS97 ist ein E-Voting-Protokoll von Cramer, Gennaro und Schoenmaker (bezeichnet auch das Dokument, welches dieses Protokoll beschreibt.) Siehe E-Voting mit homomorpher Verschlüsselung (CGS97). 12, 16, 17, 39, 61, 73, 82–87, 99, 103

Challenge-Response Ein Protokoll, in welchem ein Teilnehmer einem anderen eine gültige Antwort zu einer gestellten Frage liefern muss, um sein Wissen beweisen zu können. 24

Ciphertext Durch ein Kryptosystem verschlüsselter Text. 19, 20

Domainparameter Die 3 Domainweit definierten Grössen die ElGamal verwendet. In diesem Dokument nennen wir diese p, q und g. 17, 19

DoS Denial of Service. Boshafte Attacke auf Server-Systeme um die Verfügbarkeit des Systems zu stören. 84

ElGamal ElGamal ist ein asymetrisches Kryptosystem 1985 das von Taher Elgamal entwickelt wurde. 17, 19, 20, 25

Generator Element einer Zahlengruppe um eine weitere Gruppe mit den selben Elementen zu generieren. 17, 18

GWT Google Web Toolkit. Freies Framework von Google für die Erstellung von Web 2.0-Applikationen (http://code.google.com/webtoolkit/). 90

IDE Integrated Development Environment. Ein Anwendungsprogramm zur Entwicklung von Software, z.B. Eclipse. 90

Java Programmiersprache der Firma Oracle (http://www.oracle.com/technetwork/java/index.html). 88–90

JavaScript Programmiersprache, die in einem Webbrowser abläuft. JavaScript wird gebraucht um einer Webseite clientseitig Funktionalität zu verleihen. 89

JAX-WS Java API for XML - Web Services. Java-API zum Erstellen von SOAP-Webservices. 51

JKS Java Key Store. Ein zusammen mit der Programmiersprache Java verwendetes Format um Schlüsselmaterial und Zertifikate zu speichern. 71, 76

Mylyn Ein Plugin für Eclipse.. 91

MySQL Freie Datenbankserver-Software der Firma Oracle (http://www.mysql.com/). 89, 90

Mächtigkeit Anzahl Elemente die eine Gruppe beinhaltet. 17

Open-Audit Beschreibt zwei Eigenschaften eines Votingsystems: Jeder Wähler ist verantwortlich, seine Quittung zu überprüfen, d.h. die korrekte Verschlüsselung seiner Stimme. Jede Person, ob Wähler oder unabhängige Wahlbeobachter können das Resultat überprüfen. Wird auch "true verifiability" oder "end-to-end verifiability" genannt. 15

ORM Object-Relational Mapping oder Objektrelationale Abbildung. Ein Konzept zum Abbilden von Objekten in relationale Datenbanken. 90

Paillier Asymetrisches Kryptoverfahren (http://en.wikipedia.org/wiki/Paillier_cryptosystem). 84

PKCS12 Public Key Cryptography Standards No. 12. Dateiformat, das dazu benutzt wird, private Schlüssel mit dem zugehörigen Zertifikat passwortgeschützt zu speichern. 71, 76

PKI Public Key Infrastructure. Verwaltungssystem für digitale Zertifikate innerhalb einer Organisation. 59

Plaintext Lesbarer, unverschlüsselter Text. 19

Primzahl Eine Zahl, die nur durch sich selbst und 1 teilbar ist. 17

Private Key Privater Schlüssel in einem asymmetrischen Kryptosystem, wird verwendet um Ciphertext zu entschlüsseln oder um Daten zu signieren. 19–23

Public Key Öffentlicher Schlüssel in einem asymmetrischen Kryptosystem, wird verwendet um Daten zu verschlüsseln oder um eine Signatur zu verifizieren. 19, 20

SHA-1 Secure Hash Algorithm. Weit verbreiteter Hash Algorithmus. 73, 80

Signatur Digitale Unterschrift, mit der man den Signaturersteller identifizieren und die Datenintegrität überprüfen kann. 22

SSL/TLS Secure Socket Layer / Transport Layer Security. Standard für die Verschlüsselung von Netzwerkverbindung auf dem Netzwerk Layer 3. 59

Stützpunkt Geordnetes Paar aus Stützstelle und Stützwert. 21

Subversion Freie Software zur Versionsverwaltung von Dateien und Verzeichnissen. 87, 90, 91

SuisseID In der Schweiz eingeführter Standard für die offizielle Vergabe digitaler Zertifikate. 59

Trustee An der Entschlüsselung des Resultats beteiligte Instanz, kann als Mitglied einer Wahlbehörde betrachtet werden. 15, 85, 86, 88, 89, 92, 95–98, 102

Zero-Knowledge-Beweis Protokoll in welchem der Beweiser einen Verifizierer mit einer gewissen Wahrscheinlichkeit davon überzeugt, dass er ein Geheimnis kennt, ohne dabei Informationen über das Geheimnis selbst bekannt zu geben. 22–26, 99, 100, 102

Zertifikat Nachweis, dass der öffentliche Schlüssel eines asymmetrischen Verschlüsselungs-Verfahrens zu der vorgeblichen Person gehört. 21

Literaturverzeichnis

[1] Ronald CRAMER, Rosario GENNARO, and Berry SCHOENMAKERS. A Secure and Optimally Efficient Multi-Authority Election Scheme. 1997. 12, 14, 15, 16, 17, 39, 61, 73, 82

[2] Rolf HAENNI, Reto KOENIG, Stephan FISCHLI, and Eric DUBUIS. TrustVote: A Proposal for a Hybrid E-Voting System. 2009. 13

[3] Bruce SCHNEIER. *Angewandte Kryptographie*. Pearson Studium, 2006. 19

[4] Adi SHAMIR. How to share a secret. 1979. 21

[5] Torben PEDERSEN. A Threshold Cryptosystem without a Trusted Party. 1991. 22

[6] Peter SCHNORR. Efficient Signature Generation by Smart Cards. 1991. 25, 26

[7] David CHAUM and Torben PEDERSEN. Wallet databases with observers. 1993. 25

[8] Lee BYOUNGCHEON. Zero-knowledge proofs, digital signature variants, and their applications. 2001. 26

[9] M. Fatih KARAYUMAK. Usability Analysis and Interface Improvement of the End to End Verifiable Remote-electronic Voting System Helios. 2010. 60

[10] Jörn SCHWEISGUT. Elektronische Wahlen unter dem Einsatz kryptografischer Observer. 2007. 61

[11] R. A. PETERS. A Secure Bulletin Board. 2005. 62