Berner Fachhochschule
Technik und Informatik

Sicheres und effizientes E-Voting

Bachelor Thesis

David Berger (bergd3@bfh.ch)
Rolf Linder (lindr2@bfh.ch)

Fachbereich: Technik und Informatik
Betreuer: Prof. Dr. Rolf Haenni
Prof. Dr. Eric Dubuis
Experte: Prof. Dr. Andreas Spichiger
Datum: 20. Januar 2012

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, 20. Januar 2012

Vorname Name David Berger

Unterschrift .....

Vorname Name Rolf Linder

Unterschrift .....

Zusammenfassung

Die sichere Durchführung von elektronischen Abstimmungen ist eine der herausforderndsten Anwendungen kryptografischer Protokolle. Ein im Jahr 2005 entworfenes E-Voting-Protokoll hat unter anderem die Eigenschaft, dass der Wähler nicht erpresst werden kann. Das Protokoll besitzt allerdings auch eine markante Einschränkung bei der Stimmenauszählung. Aus diesem Grund wurden an der BFH mehrere optimierte Versionen entwickelt, welche sich diesem Problem annehmen. Der Auftrag lautete, ein Prototyp der neusten Variante zu entwickeln, um die Praktikabilität des Protokolls aufzuzeigen.

Entwickelt wurde eine auf Java RMI basierende Client-/Server-Applikation mit verschiedenen Rollen. Der Fokus bestand dabei in der Stimmenauszählung. Die Applikation beinhaltet Implementationen der Konzepte "ElGamal-Verschlüsselung", "Zero-Knowledge Proofs", "Reencryption Mix-Network" und "verteilte Berechnung und Entschlüsselung". Neben einer interaktiven Stimmabgabe wurde eine Simulation einer Abstimmung entwickelt, um Aussagen über die Leistungsfähigkeit des Protokolls machen zu können. Dabei konnte der lineare Zeitverlauf der Stimmenauszählung erfolgreich unter Beweis gestellt werden.

Im ersten Kapitel dieses Dokuments werden die Projektvorgaben sowie die Planung vorgestellt, gefolgt von einer kurzen Einführung in die Thematik von E-Voting im zweiten Kapitel. Das Kapitel drei bietet eine Einführung in die verwendeten Kryptographieteile wie z.B. die ElGamal Verschlüsselung oder Zero-Knowledge Proofs. Kapitel vier richtet sich an den am Aufbau der Applikation interessierten Leser. Das fünfte Kapitel erklärt die Benutzung der einzelnen Applikationskomponenten zur Durchführung einer Abstimmung. Die Simulation einer Abstimmung ist in Kapitel sechs beschrieben. In Kapitel sieben wird die Performance, bzw. die Ausführungszeit der entstandenen Applikation untersucht. Das achte und letzte Kapitel beinhaltet schliesslich einen Rückblick und das Fazit zur Arbeit.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Das folgende Kapitel definiert den Projektauftrag, die Bewertungs- und Abgrenzungskriterien und die damit verbundenen Ziele. Weiter werden die Projektorganisation und die Planung der Arbeit vorgestellt. Die Vorgaben aus diesem Kapitel wurden zu Beginn der Bachelor Thesis zwischen Betreuern und Studenten vereinbart. Nach Abschluss der Definition von Projektzielen und -kriterien wurden diese den Betreuern und dem Experten in Form eines Pflichtenhefts vorgestellt. Die nachfolgenden Abschnitte stützen sich auf dieses Dokument ab.

1.1 Ziele

Das Hauptziel dieser Bachelor Thesis besteht in der Realisierung eines Prototyps der E-Voting-Protokollvariante SHKS11 [6], welche im Abschnitt 2.2 vorgestellt wird. Diese verbesserte Version des E-Voting-Protokolls von Juels et al. (2005) [1] wurde an der Berner Fachhochschule entwickelt und im Paper "Efficient Vote Authorization in Coercion-Resistant Internet Voting" von Michael Schläpfer, Rolf Haenni, Reto Koenig und Oliver Spycher vorgestellt. Die Implementation soll als Proof-of-Concept Studie dienen, um die Tauglichkeit des Protokolls und seine Akzeptanz unter realen Umständen unter Beweis zu stellen. Das Protokoll soll mit Hilfe der im Modul "Projekt 2" angeeigneten Kenntnisse über die kryptographischen Komponenten so effizient wie möglich implementiert werden. Vereinfachte Ansätze oder Attrappen sind an gewissen Stellen erlaubt, um die begrenzte / vorgegebene Zeit der Bachelor Thesis einzuhalten. Die Schnittstellen, welche zur Verbindung der verschiedenen Komponenten verwendet werden, müssen entsprechend dokumentiert werden. Kryptographische Algorithmen müssen korrekt implementiert werden.

1.1.1 Musskriterien

Musskriterien sind zwingend einzuhalten und gelten als Basis für die Bewertung der Arbeit.

- Die Abstimmung muss von der Stimmabgabe bis zur Auszählung gemäss Protokoll Spezifikation von SHKS11 ablaufen. Dies beinhaltet unter anderem auch, dass die Stimme entsprechend verschlüsselt werden muss. Desweiteren müssen die zahlreichen Beweise (Zero-Knowledge Proofs), welche von den verschiedenen Parteien erbracht werden, entsprechend überprüft werden können.
- Die Implementation muss den Wahlprozess mit eingeschränkten Wahloptionen wie z.B. "ja/nein" unterstützen.

1.1.2 Sollkriterien

Sollkriterien sind weitere Ziele, welche aus Sicht Auftraggeber / Betreuer umzusetzen sind. Wenn es unter den gegebenen Vorgaben möglich ist, sollten diese ebenfalls erfüllt werden.

- Das Wahlprozedere soll so einfach wie möglich sein und kein Wissen im Bereich der Kryptographie erfordern. Die Benutzeroberfläche für die Abstimmung sowie die Administration sollen das komplexe Protokoll verbergen sofern möglich und sinnvoll. Die Applikation soll so informativ wie nötig sein, um die Korrektheit des darunterliegenden Protokolls zu belegen.
- Es soll möglich sein, eine (fiktive) Wahl zu simulieren und die diversen Teile des Protokolls automatisiert auszuführen. Dabei soll festgestellt werden, ob die Applikation allfälligen Wahlbetrug (z.B. durch das Benutzen von falschen Credentials) erkennt und verhindert.
- Als Proof-of-Concept Studie soll die Applikation Aussagen darüber liefern, wie viele Stimmen in nützlicher Frist überprüft werden können.
- Client- und Server-Teil sollen getrennt werden.
- Die Entschlüsselung der Stimmen soll im Distributed Verfahren erfolgen.

1.1.3 Kannkriterien

Kannkriteren definieren optionale Erweiterungen, welche allenfalls zusätzlich umgesetzt werden können.

- Einsatz von Threshold Decryption. In diesem Kontext würde dies bedeuten, dass eine Abstimmung auch stattfinden kann, wenn sich nur ein Teil aller Parteien beteiligt oder einzelne Parteien nicht vertrauenswürdig wären.
- Unterstützung zusätzlicher Wahloptionen anstelle von nur "ja" und "nein".
- Zusätzliche Absicherung des Bulletin Boards (der Ort, an dem sämtliche Wahlinformationen veröffentlicht werden), z.B. durch die Sicherstellung des "append-only" Modus über Hashes und Signaturen oder die Verteilung des Bulletin Boards auf mehrere Parteien.
- Authentifizierung der Teilnehmer (z.B. der Mixer). Dadurch würde z.B. sichergestellt, dass es sich wirklich um den legitimen Mixer handelt und, dass dieser nicht bereits gemixt hat.

1.1.4 Abgrenzungskriterien

Dieser Abschnitt beschreibt die Abgrenzungen, welche seitens der Betreuer zu Beginn der Arbeit festgelegt wurden.

- Der Hauptfokus der entstehenden Applikation soll auf dem Teil der Stimmenauszählung ("Vote Authorization") liegen. Die Registrierung wird daher stark vereinfacht. So wird vorausgesetzt, dass sich der Wähler bei der Wahlbehörde vorgängig registriert und einen digitalen Stimmrechtsausweis erhält. Dieser Ausweis benutzt er anschliessend für die Authentisierung an der E-Voting-Applikation. Details siehe Anhang A.4.
- Das Bulletin Board muss nicht alle Anforderungen erfüllen. Anstelle von verteilten Systemen reicht eine zentrale Datenbank aus. Das Board soll jedoch unterschiedliche Typen von Einträgen unterstützen (Wahlregistereintrag, abgegebene Stimmen, Ergebnisse der Mixer, etc.), der Eintrag selber sollte daher möglichst generisch sein. Grundsätzlich soll die Applikation nur die Operationen "read" und "append" anbieten. Es dürfen also keine Einträge gelöscht werden.
- Für das Mixing existiert bereits eine Implementierung, welche möglicherweise eingesetzt werden kann. Verificatum von Douglas Wikstroem ist in Java geschrieben, liefert eine Iteration ("Shuffle") und die entsprechenden Zero-Knowledge Proofs dazu. Es muss allerdings geprüft werden, ob die Implementierung die Verschlüsselung von mehreren ElGamal Tupeln pro Stimme zulässt.
- Anstelle eines Threshold Verfahrens wird ein Distributed Verfahren angewendet. Das heisst, es kann davon ausgegangen werden, dass alle benötigten Parteien an der Wahl teilnehmen und vertrauenswürdig sind.
- Ob die Applikation web-basiert oder aber als eigenständige Client/Server-Anwendung realisiert werden soll, wird den Studenten überlassen. Eine browserbasierte Lösung kann erfahrungsgemäss zu Problemen im Bereich der Performance führen, zudem besitzt JavaScript Limitationen im Bereich von Datentypen und Rechenoperationen.
- Die Client-Seite soll sich auf die funktionale Ebene reduzieren, die "Usability" steht in diesem Projekt im Hintergrund.

1.2 Produkt

Aufgrund der vorangehend genannten Ziele und Kriterien wurde die nachfolgende Beschreibung erstellt, welche in die Bereiche Anwendung, Zielgruppen, Produktdetails und Qualitätsanforderungen aufgeteilt ist.

1.2.1 Anwendungsbereiche

Das Produkt (die Applikation) soll zu Forschungs- und Studienzwecken dienen und wird nicht für den produktiven Einsatz konzipiert. Der Anwendungsbereich beschränkt sich daher auf ein Proof-of-Concept für die an der Berner Fachhochschule entwickelte Protokoll Variante.

1.2.2 Zielgruppen

Zielpersonen dieses Projekts sind in erster Linie die Betreuer sowie der Experte. Im Kontext des E-Voting-Systems selbst existieren, abgesehen von den Parteien, welche sich an der Stimmenauszählung beteiligen, grundsätzlich folgende drei Zielgruppen:

- **Wähler** - nutzt Applikation für die Stimmabgabe
- **Administrator** - kann neue Abstimmungen einrichten und bestehende verwalten
- **Beobachter** - hat die Möglichkeit, sämtliche Prozesse innerhalb der Applikation auf ihre Richtigkeit zu überprüfen

1.2.3 Produktfunktionen

- **Registrierung eines neuen Wählers:** Der Wähler kann sich auf der Wahlplattform registrieren und erhält, nach Überprüfung seiner Identität, einen digitalen Stimmrechtsausweis.
- **Abstimmung einrichten:** Der Administrator kann eine neue Abstimmung einrichten und die entsprechenden Wahlmöglichkeiten spezifizieren.
- **Abgabe einer Stimme:** Der Wähler gibt seine Stimme ab, welche gemäss Protokoll Spezifikation nach dem ElGamal-Verfahren verschlüsselt und auf dem Bulletin Board gespeichert wird, wo sämtliche Stimmen anonymisiert einsehbar sind. Eine doppelte Stimmabgabe muss explizit möglich sein, resp. soll nicht verhindert werden.
- **Stimmenauszählung:** Die Abstimmung wird geschlossen (keine weitere Stimmabgabe mehr möglich). Anschliessend werden gemäss der Protokoll Spezifikation ungültige Stimmen ermittelt und eliminiert. Am Ende werden sämtliche gültigen Stimmen von den Wahlbehörden gemeinsam entschlüsselt, summiert und das Resultat der Abstimmung auf dem Bulletin Board abgelegt.

Für eine detaillierte Beschreibung der einzelnen Funktionen wird an dieser Stelle auf die Use Cases im Anhang A verwiesen.

1.2.4 Produktdaten

Sämtliche applikationsrelevanten Daten werden in einer Datenbank gespeichert. Im Gegensatz zu den Benutzerdaten sollen abstimmungsbezogene Daten weder vom Benutzer noch vom Administrator nachträglich geändert werden können. Da es sich um eine Machbarkeitsstudie handelt, besteht für Tests jedoch die Möglichkeit, Anpassungen direkt in der Datenbank vorzunehmen.

1.2.5 Qualitätsanforderungen ür die Entwicklung

Nachfolgende Kriterien werden zur Erhöhung bzw. Erreichnung der gewünschten Qualität aufgestellt:

- Einhaltung der Protokoll Spezifikation
- Versionierung und Sicherung sämtlicher projektspezifischer Daten
- Einsatz von etablierten und aktuellen Entwicklungsumgebungen sowie Frameworks
- ausreichend kommentierter und nachvollziehbarer Programmcode

1.3 Technische Produktumgebung

Die nachfolgenden Abschnitte beschreiben die Rahmenbedingungen bezüglich Software, Hardware, Schnittstellen und der Entwicklungsumgebung.

1.3.1 Software

Die Applikation wird in der Programmiersprache Java codiert und als Projekt auf der Plattform Origo[1] geführt. Im Rahmen der Initialisierungsphase wurde entschieden, anstelle einer Web-Applikation eine Client-/Server-Applikation zu entwickeln. Sowohl Client als auch Server müssen über ein Java Runtime Environment (JRE) verfügen. An den Client werden keine weiteren Anforderungen gestellt. Serverseitig wird eine MySQL Datenbank verwendet, wobei diese mit relativ geringem Aufwand gegen ein anderes Produkt ausgetauscht werden kann. Für einzelne Applikationsteile werden spezifische Frameworks eingesetzt, welche im Abschnitt 4.1 beschrieben werden.

1.3.2 Hardware

An die Hardware werden grundsätzlich keine speziellen Anforderungen gestellt. Die Applikation soll so ausgelegt werden, dass sie grundsätzlich server- und nicht clientseitig Ressourcen beansprucht, um die Nutzung einer möglichst breiten Gerätebasis zu ermöglichen.

1.3.3 Schnittstellen

Eine Schnittstelle zu anderen Applikationen ist in diesem Projekt nicht vorgesehen. Die Applikation soll aber explizit so entwickelt werden, dass die Entwicklung von weiteren Modulen oder Schnittstellen zu anderen Applikationen ermöglicht wird. Eine Integration der Java Library "Verificatum" als Reencryption Mix-Network wird im Rahmen dieser Arbeit geprüft.

1.3.4 Entwicklungsumgebung

Während der Umsetzung dieser Arbeit werden die nachfolgenden Werkzeuge eingesetzt:

- **Integrated Development Environment (IDE):**
  Eclipse (Lizenz: Eclipse Public License), `http://www.eclipse.org/`
- **Projektverwaltung:**
  Origo Plattform (Lizenz: MIT License), `http://evoting-thesis-lb12.origo.ethz.ch`
- **UML Digramme:**
  ArgoUML (Lizenz: Eclipse Public License), `http://argouml.tigris.org/`
- **Dokumentation, Grafiken und Präsentation:**
  LaTeX (Texmaker und TeX Live 2011, beide unter GPLv2+ lizenziert)
  `http://www.xm1math.net/texmaker/` und `http://www.tug.org/texlive/`

1.4 Organisation

Die Autoren dieser Bachelor Thesis studieren Informatik mit dem Schwerpunkt "IT Security" im 9. Semester an der Berner Fachhochschule, Abteilung Technik und Informatik. Betreut werden sie von zwei Dozenten, welche die Forschungsgruppe E-Voting an der Berner Fachhochschule leiten. Abbildung 1.1 zeigt die Projektorganisation dieser Arbeit.

1.5 Projektplan

Innerhalb der ersten zwei Semesterwochen wurde ein Projektplan für die Umsetzung der Arbeit erstellt. Das Projekt beginnt am 19. September 2011 (KW 38) und endet am 20. Januar 2012 (KW 3) mit der Abgabe dieser Dokumentation. Zwei Wochen später erfolgt die Präsentation der Erkenntnisse in Form der Verteidigung der Bachelor Thesis. Abbildung 1.2 zeigt den geplanten Verlauf der Arbeiten während dieses Zeitraumes.

[1] Origo ist eine Plattform zur Code- / Projektverwaltung der ETH Zürich `http://www.origo.ethz.ch/`

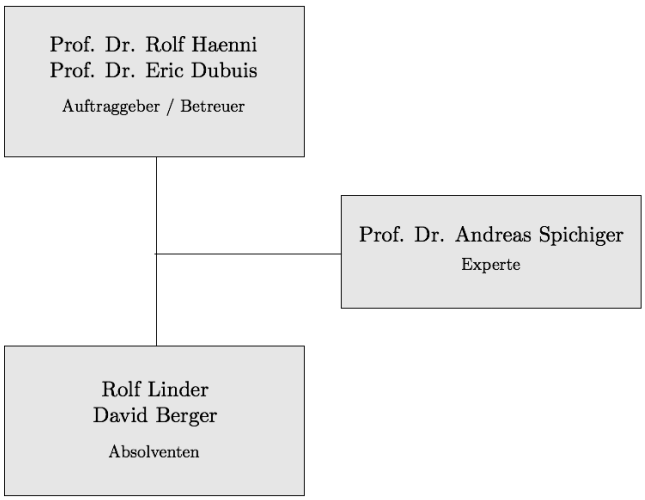


Abbildung 1.1: Projektorganisation

Abbildung 1.2: Projektplanung

2 E-Voting

Das Thema E-Voting stösst seit einigen Jahren auf reges Interesse. Die weltweite Verbreitung des Internets eröffnet unter anderem auch neue Möglichkeiten im Bereich der Interaktion zwischen Bürger und Staat. Wie die Zeitschrift Netzwoche im Oktober 2011 berichtete[8], beteiligen sich in der Schweiz mittlerweile 13 Kantone an Pilotprojekten im Bereich E-Voting. Neben den Pionierkantonen Genf, Neuenburg und Zürich sind dies Aargau, Bern, Basel, Freiburg, Graubünden, Luzern, Schaffhausen, Solothurn, Thurgau und St. Gallen. Von besonderem Interesse sind dabei die knapp 700'000 im Ausland lebenden Bürger, da die Zustellung der Wahlunterlagen bei ihnen oft verspätet erfolgt und jährlich Kosten in Millionenhöhe generiert.

Wie erste Erfahrungen zeigten, erfreut sich E-Voting grosser Beliebtheit: Im Rahmen eines Pilotversuchs konnten im Mai 2011 erstmals rund 2'600 Auslandschweizer elektronisch über das Energiegesetz im Kanton Bern abstimmen. Die Stimmbeteiligung lag mit 61,9% deutlich höher als bei herkömmlichen Abstimmungen und 98% der Nutzer gaben an, in Zukunft wieder elektronisch abzustimmen. Bei den letzten National- und Ständeratswahlen vom 23. Oktober 2011 wurde zum ersten Mal die elektronische Stimmabgabe auch für landesweite Wahlen ermöglicht. Gemäss dem Nachrichtenportal swissinfo.ch [10] fiel die Bilanz auch hier überwiegend positiv aus: Insgesamt nutzten 3'562 Personen das Angebot, was mehr als der Hälfte der abstimmenden Auslandschweizer der teilnehmenden Kantone entspricht. Sowohl die Bundeskanzlei als auch die teilnehmenden Kantone zeigten sich äusserst zufrieden und sprachen von einem reibungslosen Ablauf. Der Bund sieht sich daher in seinen Bestrebungen bekräftigt, immer mehr Auslandschweizern die Stimmabgabe per Internet zu ermöglichen.

Ähnlich wie bei herkömmlichen Abstimmungen bestehen im Bereich von elektronischen Abstimmungen diverse Herausforderungen. So muss beispielsweise sichergestellt werden, dass nur berechtigte Personen abstimmen können und, dass es sich sowohl beim Wähler als auch bei der Wahlbehörde um die korrekte Person handelt. Die Auszählung der Stimmen muss transparent und für alle nachvollziehbar erfolgen, darf jedoch keinen Aufschluss darüber geben, wer wie abgestimmt hat. Das E-Voting-System muss absolut vertrauenswürdig sein, insbesondere wenn eine Abstimmung ein überraschendes Result liefert. Es sei hier angemerkt, dass die heute im Einsatz stehenden Pilotsysteme einige der Anforderungen, welche die Forschung an E-Voting-Systeme stellt, noch nicht oder nur teilweise erfüllen.

2.1 Das Basisprotokoll: JCJ05

Das im Jahr 2005 von Ari Juels, Dario Catalano und Markus Jakobsson entworfene E-Voting-Protokoll [1] (nachstehend "JCJ05" genannt) entspricht dem neusten Stand der Technik und hat bemerkenswerte Sicherheitseigenschaften. Im Zentrum des Protokolls steht die sogenannte "Coercion Resistancy", also die Eigenschaft, dass die Stimme von Wählern nicht erpressbar ist. So soll es einem Coercer[1] nicht möglich sein, Wähler dazu zu zwingen, auf eine bestimmte Art und Weise zu stimmen, nicht an der Wahl teilzunehmen oder gar ihre geheimen Schlüssel zu veröffentlichen.

Die Autoren nehmen Bezug auf frühere Arbeiten und erwähnen in diesem Zusammenhang insbesondere die Eigenschaft der "receipt-freeness", was soviel bedeutet wie, dass ein Wähler dem Coercer nicht beweisen kann, wie er gestimmt hat, selbst wenn er das wollte. Doch das Erfüllen der Eigenschaft "receipt-freeness" erlaubt weiterhin die folgenden Attacken:

- **Randomization Attack:** Der Coercer zwingt den Wähler, zufällig generiertes Wahlmaterial einzureichen. Unter Umständen sind weder Coercer noch Wähler in der Lage festzustellen, wie gestimmt wurde. Das Ziel dieser Attacke besteht darin, die Stimme des Wählers aufgrund der Zufälligkeit zunichte zu machen.
- **Forced-Abstention Attack:** Der Wähler wird gezwungen, sich seiner Stimme komplett zu enthalten. Da die meisten bisherigen E-Voting-Systeme die Wähler direkt authentisieren. kann der Coercer feststellen, ob ein Wähler seine Stimme abgegeben hat und entsprechend Druck ausüben.

[1] eine Person, welche einen Wähler erpressen will

- **Simulation Attack:** Die Eigenschaft der "receipt-freeness" hindert den Wähler leider nicht daran, seinen privaten Schlüssel weiterzugeben oder zu verkaufen, wodurch der Coercer eine fremde Stimmabgabe "simulieren" kann.

Die Antwort der Autoren ist dementsprechend eine stärkere Eigenschaft als "receipt-freeness", genannt "coercion-resistant", welche auch die drei genannten Attacken verhindert. Weiter werden die Eigenschaften "Korrektheit" und "Überprüfbarkeit" gefordert und entsprechend formal definiert. Ein E-Voting-System nach JCJ05 fordert lediglich einen anonymen Kanal für die Stimmabgabe, im Gegensatz zu bestehenden E-Voting-Verfahren, welche von einem abhörsicheren Kanal ausgehen. Ein abhörsicherer Kanal wird im System nach JCJ05 ausschliesslich für die Registrierung der Wähler (einmalig) gefordert, wobei hier der Postweg als Alternative genannt wird.

Die Idee bei JCJ05 besteht darin, dass die Identität des Wählers während des Stimmprozederes geheim bleibt und stattdessen die Gültigkeit der Stimme mit einer Wählerliste (genannt Voter Roll) verglichen wird. Jeder Wähler erhählt dazu ein privates Credential (genannt σ), welches nur der Wähler selbst kennt. Im Falle einer Erpressung, kann der Wähler dem Coercer anstelle des gültigen Credentials σ ein vermeintlich gültiges $\tilde{\sigma}$ aushändigen bzw. zur Stimmabgabe verwenden. Weder Wähler noch Coercer können die die Gültigkeit, resp. Ungültigkeit von $\tilde{\sigma}$ beweisen. Der Wähler hat jedoch die Möglichkeit, nach der (ungültigen) Stimmabgabe mit $\tilde{\sigma}$ noch eine gültige Stimmabgabe mit dem korrekten Credential σ durchzuführen.

Das durch JCJ05 definierte System besitzt verschiedene Rollen, welche in der Abbildung 2.1 dargestellt sind. Der Wähler (V) interagiert mit der Wahlbehörde (R) und dem öffentlichen Bulletin Board (BB). Die beiden Rollen (bzw. Entitäten) Mixing Authority (M) und Tallying Authority (T) werden nach der Stimmabgabe durch die Wähler für die Anonymisierung bzw. Auszählung der Stimmen benötigt.

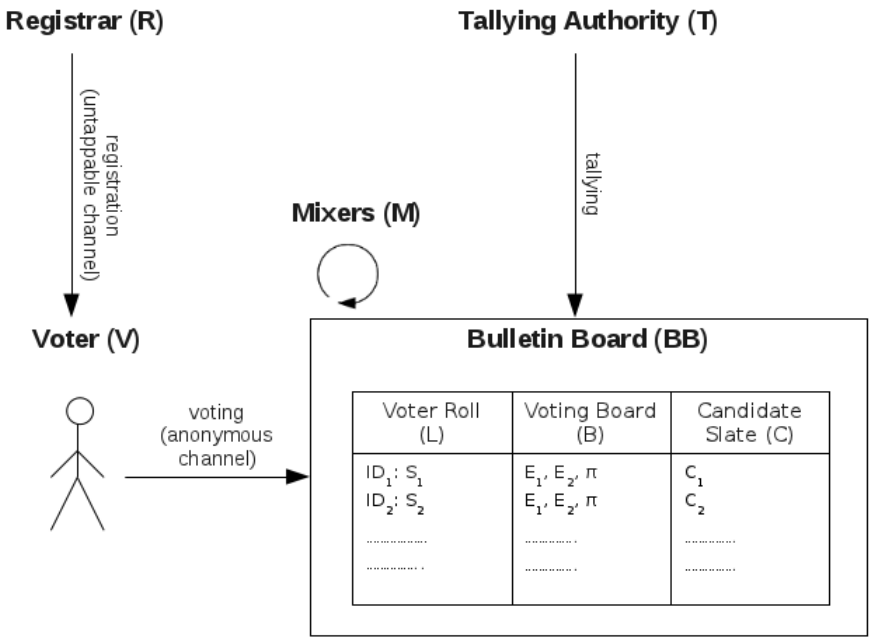


Abbildung 2.1: Rollen innerhalb eines E-Voting-Systems nach JCJ05

Das Bulletin Board ist aufgeteilt in die drei folgenden Bereiche:

- Voter Roll (L) enthält Angaben zum Wähler (ID_i) und sein vom Registrar verschlüsselter Stimmrechtsausweis (S_i)
- Voting Board (B) enthält der vom Wähler verschlüsselte Stimmrechtsausweis σ und die verschlüsselte Wahl (E_1, E_2) sowie die entsprechenden Beweise (Π)
- Candidate Slate (C) enthält die möglichen Wahloptionen zur Abstimmung

An dieser Stelle ist anzumerken, dass die Rollen R, T und M im Sinne einer Gewaltentrennung auf mehrere Systeme / Personen aufgeteilt werden. Um dies zu erreichen, werden die Techniken "Secret Sharing" und "Threshold Decryption" eingesetzt. Auf die einzelnen Use Cases von JCJ05 wird an dieser Stelle nicht weiter eingegangen, da für dieses Projekt grundsätzlich nur diejenigen des Protokolles SHKS11 relevant sind.

2.2 Eine optimierte Variante: SHKS11

JCJ05 konnte im Bereich der Sicherheit von E-Voting-Protokollen neue Massstäbe setzen, besitzt allerdings eine Limitation bei der Stimmenauszählung: Um doppelte und gefälschte Stimmen zu ermitteln, wird jede Stimme mit jeder anderen Stimme verglichen. Dies führt dazu, dass die Zeit für die Überprüfung der Stimmen quadratisch mit der Anzahl Stimmen ansteigt. Bereits ab einer vergleichsweise kleinen Anzahl von 100 Stimmen dauert die Auszählung so mehrere Stunden, was in der Praxis nicht praktikabel ist. Aus diesem Grund wurden an der Berner Fachhochschule mehrere Varianten entwickelt, welche sich dem Problem der quadratischen Ausführungszeit der Stimmenauszählung annehmen. Die aktuellste Variante wurde im Sommer 2011 von Michael Schläpfer, Rolf Haenni, Reto Koenig und Oliver Spycher unter dem Namen "Efficient Vote Authorization in Coercion-Resistant Internet Voting" [6] publiziert und wird im weiteren Verlauf dieses Dokuments mit "SHKS11" referenziert.

SHKS11 ist stark angelehnt an JCJ05. Im Bereich der "Vote Authorization", also der Stimmenüberprüfung, gibt es allerdings folgende Unterschiede:

- Um Duplikate zu entfernen, wird das von Smith und Weber[9, 11, 12] vorgestellte Verfahren angewendet, welches eine lineare Durchführungszeit zur Erkennung von mehrfachen Stimmabgaben ermöglicht. Anstatt wie bei JCJ05 ein Plaintext Equivalence Test (PET) über sämtliche abgegebene Stimmen durchzuführen, wird jede der Verschlüsselungen c_i $(1 \leq i, j \leq n)$ mit einem Zufallswert z potenziert und anschliessend entschlüsselt:

  $$c_1{}^z = Enc_\epsilon(\sigma_1{}^z), \ldots, c_n{}^z = Enc_\epsilon(\sigma_n{}^z)$$

  Es sei angemerkt, dass z nur den Tallying Authorities bekannt ist und verteilt generiert wird. Die verteilt entschlüsselten Blindwerte $\sigma_i{}^z$ werden in einer Hashtabelle abgelegt und dabei implizit miteinander verglichen (eine Hashtabelle erlaubt keine Duplikate). Da stets mit dem gleichen Zufallswert z verblindet wurde, gilt $\sigma_i = \sigma_j$ wenn $\sigma_i{}^z = \sigma_j{}^z$.

- Sowohl SHKS11 als auch JCJ05 sehen vor, dass der Wähler absichtlich gefälschte Stimmen generiert. Zu diesem Zweck spezifiziert der Wähler bei SHKS11 ein sogenanntes Anonymitätsset, wobei der Parameter β die minimale Grösse der absichtlich gefälschten Stimmen festlegt. Beträgt β beispielsweise 30, so bedeutet dies, dass neben der eigentlichen Stimme absichtlich mindestens 29 gefälschte Stimmen für die im Anonymitätsset mitgelieferten Identitäten generiert werden.

  Die maximale Anzahl gefälschter Stimmen wird durch die Grösse des Stimmregisters (Voter Roll) festgelegt. Das "Worst Case Szenario" im Bezug auf die Aufwändigkeit des Protokolls wäre demnach, wenn als Grösse des Anonymitätssets zufällig die Grösse des Wahlregisters gewählt wird. In diesem Fall entspricht die Ermittlung der gefälschten Stimmen der gleichen Komplexität wie beim Originalprotokoll JCJ05.

  Sämtliche der so generierten Stimmen werden nach dem Replikationsvorgang einem Reencryption Mix-Network übergeben, welches als Eingabe die drei Tupel (A, B, S_i) entgegennimmt und als Ausgabe drei Tupel (A', B', S_i') liefert. Konkret findet eine Wiederverschlüsselung der ElGamal Tupel sowie eine Vertauschung der Reihenfolge statt. Im Anschluss führt die Wahlbehörde ein PET über die Tupel (A', S_i') durch. Liefert der Test "false" zurück, wurde die Stimme mit einer falschen Identität abgegeben. Die Stimme wird in diesem Fall als ungültig markiert und von der Zählung ausgeschlossen. Zuletzt werden die noch gültigen Tupel B' gemeinsam entschlüsselt und ausgezählt.

Die genaue Funktionsweise des Protokolls und seiner Akteure wird in Kapitel 4 ausführlich erklärt. Beide der oben beschriebenen Vorgänge besitzen eine Komplexität von $O(n)$ und verleihen dem Protokoll SHKS11 somit ein lineares Laufzeitverhalten für die Stimmenauszählung. Dieses Verhalten soll anhand der Prototyp-Implementation praktisch aufgezeigt werden. Nachfolgend ist bei der Nennung von "Protokoll" oder "Spezifikation" die Variante SHKS11 gemeint, sofern nicht anders vermerkt.

3 Kryptographische Grundlagen

Bei der Anwendung von Protokollen im Bereich von E-Voting kommen verschiedene kryptographische Algorithmen zum Einsatz. Die nachfolgenden Grundlagen wurden innerhalb des Moduls "7302, Projekt 2" durch die beiden Studierenden als Vorbereitung für die Bachelor Thesis erarbeitet. Die Einführung ist nicht abschliessend und setzt Grundkenntnisse in der Mathematik sowie im Umgang mit kryptographischen Algorithmen voraus.

Im Verlauf dieses Berichts arbeiten wir mit mathematischen Gruppen, genauer gesagt mit multiplikativen, zyklischen Gruppen. Die Gruppen werden dabei mit Hilfe einer Primzahl der Form $p = kq + 1$ (p, q sind grosse Primzahlen und $k \in \mathbb{Z} \geq 2$) gebildet und wir verwenden wie üblich einen Generator g der Ordnung q. Wir bezeichnen eine multiplikative Gruppe modulo p mit ${\mathbb{Z}_p}^*$. Eine Gruppe generiert aus dem Generator g bezeichnen wir mit $< g >= \mathcal{G}_q \subseteq {\mathbb{Z}_p}^*$.

Diskreter Logarithmus: Unter dem *diskreten Logarithmus* [4] versteht man das Lösen der Gleichung $g^x \equiv y \mod p$, bei gesuchtem Wert x und gegebenen Werten g, y und p (wobei p eine Primzahl ist, $g, y \in \mathcal{G}_q$ und $x \in \mathbb{Z}_q$). Das Berechnen des Werts y ist einfach. Umgekehrt gilt die Berechnung des diskreten Logarithmus in multiplikativen Gruppen ohne Kenntnis von x jedoch bis heute als aufwändig.

Unter der Voraussetzung, dass es sich um eine zyklische Gruppe mit einer sehr grossen Ordnung handelt, geht man aktuell davon aus, dass keine einfache Berechnung des diskreten Logarithmus existiert. Dieser Fakt wird mit dem Begriff "Diskretes-Logaritmus-Problem" (oder abgekürzt DLP) umschrieben. Neben dem naiven Ansatz zur Lösung des DLPs (durchprobieren aller x in $\mathbb{Z}_q$) existieren optimierte Algorithmen wie z.B. "Babystep-Giantstep" [3], welche jedoch bei entsprechend gross gewählten Gruppen keine effiziente / einfache Berechnung des diskreten Logarithmus erreichen.

3.1 Die ElGamal Verschlüsselung

Das ElGamal Kryptosystem [5] ist ein asymmetrisches Verschlüsselungsverfahren, das durch die Anwendung einer sogenannten *Randomisierung* die mehrmalige Verschlüsselung desselben Klartextes in unterschiedliche Chiffretexte umwandelt. Das System wird in vier Phasen unterteilt: der Aufbau, die Schlüsselgenerierung, die Verschlüsselung und die Entschlüsselung.

Aufbau: Einmalig vor der Benutzung des ElGamal Kryptosystems werden die öffentlichen Parameter bestimmt. Diese umfassen die Gruppenordnungen p, q einer ElGamal Instanz, sowie den Generator g. Die Wahl der Gruppe erfolgt über die Wahl einer grossen Primzahl der Form $p = kq + 1$ (p, q sind Primzahlen). Wie bereits beschrieben, wird durch die Wahl des entsprechenden Generators g eine Gruppe $\mathcal{G}_q$ der Ordnung q gebildet. Damit sind die Werte p, q (Gruppen-Parameter) und g (Generator) definiert und der Aufbau abgeschlossen.

Schlüsselgenerierung: Es wird ein Schlüsselpaar generiert, damit Nachrichten chiffriert werden können. Dies erfolgt durch die zufällige Auswahl eines Elements x aus der Menge $\mathbb{Z}_q$ ($x \in_R \mathbb{Z}_q$). Das Element x ist privat und geheim zu halten (privater Schlüssel). Zum geheimen Element x berechnen wir den öffentlichen Schlüssel $y \in \mathcal{G}_q$ mit $g^x \mod p$. Der öffentliche Schlüssel wird zusammen mit den Werten aus der vorherigen Berechnung (p, q und g) veröffentlicht. Damit ist die Grundlage gelegt, um Nachrichten zu verschlüsseln.

Verschlüsselung einer Nachricht: Die Nachricht m ($m \in \mathcal{G}_q$) im obigen definierten ElGamal System soll verschlüsselt werden. Dazu wird als erstes zufällig eine Randomisierung r aus der Menge $\mathbb{Z}_q$ ($r \in_R \mathbb{Z}_q$) gewählt und der Wert $g^r \mod p$ berechnet. Die chiffrierte Nachricht ist zusammengesetzt aus dem Wert g^r und dem Wert $y^r \cdot m$. Auch diese Werte werden modulo p gerechnet, wie alle weiteren Berechnungen. Aus Gründen der Einfachheit wird diese Nennung fortan weggelassen. Es entsteht somit das *ElGamal Tupel*

$c = (g^r, y^r \cdot m)$. Durch die Verwendung der Randomisierung resultiert bei der Verschlüsselung einer Nachricht m mit dem ElGamal System bei jedem Durchgang (jeder Verschlüsselung) ein anderes ElGamal Tupel (anderer Ciphertext).

Entschlüsselung einer Nachricht: Um eine verschlüsselte Nachricht zu dechiffrieren, benötigt man den zum öffentlichen Schlüssel y entsprechenden geheimen Schlüssel x.[1] Anschliessend können wir m mit Hilfe der Gleichung

$$m = \frac{y^r \cdot m}{(g^r)^x}$$

berechnen. Da der öffentliche Schlüssel y dem Wert g^x entspricht und die Potenzgesetze auch in multiplikativen Gruppen gültig bleiben, ist y^r faktisch gleich $(g^x)^r = g^{x \cdot r} = (g^r)^x$. Man benötigt das multiplikativ inverse Element von $(g^r)^x$ (da die Division innerhalb von multiplikativen Gruppen durch die Multiplikation mit dem inversen Element berechnet wird), um die Nachricht m zu entschlüsseln.

Das ElGamal Verschlüsselungssystem besitzt eine weitere wichtige Eigenschaft: es ist *homomorph*. Ein homomorphes Verschlüsselungssystem erlaubt es, durch die Anwendung von Operationen auf die Chiffrate (ElGamal Tupel) eine Veränderung auf den zugrundeliegenden Klartexten erreicht wird. Das ElGamal System ist multiplikativ homomorph, das bedeutet, durch die Multiplikation zweier ElGamal Tupel $c = (a, b) = (g^r, y^r \cdot m)$ multiplizieren sich die Werte g^r und $y^r \cdot m$ der beiden ElGamal Tupel miteinander. Wenden wir diese Technik auf die ElGamal Tupel $c_1 = (g^{r_1}, y^{r_1} \cdot m_1)$ und $c_2 = (g^{r_2}, y^{r_2} \cdot m_2)$ an und multiplizieren $c_1 \cdot c_2$ resultiert daraus $c_{12} = (g^{r_1+r_2}, y^{r_1+r_2} \cdot m_1 \cdot m_2)$, eine Verschlüsselung der Nachricht $m_1 \cdot m_2$.

3.2 Zero-Knowledge Proof

Mit Hilfe eines *Zero-Knowledge Proof*s ist es möglich, die Kenntnis eines Geheimnisses zu beweisen, ohne das Geheimnis selbst zu enthüllen. Wird der Beweis nicht-interaktiv geführt, sprechen wir von einem "Non-Interactive Zero-Knowledge Proof", abgekürzt "NIZKP". Im weiteren Verlauf dieses Kapitels nutzen wir nicht-interaktive Varianten, also NIZKPs. Die Beweisführung mit den Akteuren *Prover* (derjenige, der etwas beweisen will) und dem *Verifier* (derjenige, der den Beweis verifizieren will) erfolgt nach einem sogenannten Σ-Protokoll, wobei die Werte t (das Commitment), c (die Challenge) und s (die Response) zur Anwendung kommen. Nachfolgend werden die in diesem Projekt verwendeten Beweistypen vorgestellt.

Proof of Knowledge of Discrete Logarithm: Will man beweisen, dass man den diskreten Logarithmus x zum Wert $y = g^x$ in der Gruppe $\mathcal{G}_q$ mit dem Generator g kennt, wendet man den "Proof of Knowledge of Discrete Logarithm" an. Die Beweisführung nach dem oben beschriebenen Ablauf (t,c und s) für den Beweis des diskreten Logarithmus wird als Schnorr-Protokoll [7] bezeichnet.

Im ersten Schritt wählt der Prover zufällig eine Zahl ω aus der Menge $\mathbb{Z}_q$ ($\omega \in_R \mathbb{Z}_q$) aus. Damit berechnet er den Wert t mit $t = g^\omega$. Als nächstes bildet er mit Hilfe einer kryptographischen Hash-Funktion H den Hash-Wert c von t und y, also $c = H(t, y)$ und berechnet zuletzt s mit $s = \omega + x \cdot c$. Die Werte t, c und s werden nun veröffentlicht.

Ein Verifier prüft, ob die Gleichungen

$$t \cdot y^c \stackrel{?}{=} g^s \tag{3.1}$$

$$c \stackrel{?}{=} H(t, y) \tag{3.2}$$

erfüllt sind.

Die formale Darstellung dieses Beweises lautet $ZKP\{(x) : \log_g y = x\}$.

[1] Es ist ebenfalls möglich mit Kenntnis der Randomisierung r zu entschlüsseln (ohne Kenntnis des privaten Schlüssels). Damit berechnet sich die Nachricht m aus
$m = \frac{y^r \cdot m}{y^r}$

Proof of Knowledge of Plaintext or Randomness: Auf Basis des "Proof of Knowledge of Discrete Logarithm" ist es möglich, die Kenntnis des Klartextes bzw. der Randomisierung zu beweisen.

Wie bereits erwähnt, kann eine Entschlüsselung mit Hilfe des privaten Schlüssels x oder durch Kenntnis der Randomisierung r vorgenommen werden. Daraus leitet sich ab, dass mit dem Beweis der Kenntnis von r implizit bewiesen werden kann, dass auch die Nachricht selbst bekannt ist.

Bezogen auf das ElGamal Tupel $c = (a, b) = (g^r, y^r \cdot m)$ wenden wir den Zero-Knowledge Proof

$$ZKP\{(r) : \log_g a = r\}$$

an.

Proof of Knowledge of Equality of Discrete Logarithm: Soll bewiesen werden, dass der diskrete Logarithmus zweier Elemente übereinstimmt, wenden wir den Beweis "Equality of Discrete Logarithm", formal

$$ZKP\{(x) : \log_g y_1 = x \wedge \log_h y_2 = x\}$$

an. Dabei sind g und h zwei unterschiedliche Generatoren. Die hier verwendete Beweisführung wurde in [2] vorgestellt.

Der Prover wählt erneut zufällig $\omega \in_R \mathbb{Z}_q$, berechnet zwei Commitments $t_1 = g^\omega$ und $t_2 = h^\omega$ und berechnet den Hashwert $c = H(t_1, t_2, y_1, y_2)$. Zuletzt wird $s = \omega + x \cdot c$ berechnet und die Werte t_1, t_2, c und s werden veröffentlicht.

Ein Verifier prüft, ob die Gleichungen

$$t_1 \cdot {y_1}^c \stackrel{?}{=} g^s \tag{3.3}$$

$$t_2 \cdot {y_2}^c \stackrel{?}{=} h^s \tag{3.4}$$

$$H(t_1, t_2, y_1, y_2) \stackrel{?}{=} c \tag{3.5}$$

erfüllt sind.

Proof of Validity: Um zu beweisen, dass eine Verschlüsselung einen Wert aus einer Liste von n gültigen Werten darstellt, bedarf es eines etwas komplizierteren Beweises, welcher an dieser Stelle einfachheitshalber nur für $n = 2$ beschrieben wird. Eine ausführliche, formale Beschreibung des Beweises für ein beliebiges n findet sich im Anhang B.

Eine Verschlüsselung besteht aus einem ElGamal Tupel $c = (a, b) = (g^r, y^r \cdot m)$, wobei $m \in \{m_1, m_2\}$ gilt. Die Liste der möglichen Verschlüsselungen lautet somit:

$$c_1 = (g^r, y^r \cdot m_1), \quad c_2 = (g^r, y^r \cdot m_2)$$

Will man beweisen, dass die Verschlüsselung c eine gültige Verschlüsselung vom Wert m ($m \in m_1, m_2$) ist, nutzt man eine Verkettung mehrerer "Proof of Knowledge of Equality of Discrete Logarithm" und beweist

$$(\log_g a = \log_y b_1) \text{ OR } (\log_g a = \log_y b_2),$$

wobei

$$b_1 = \frac{b}{m_1}, \quad b_2 = \frac{b}{m_2}$$

gilt.

Da der Prover diesen Beweis nur für ein einzelnes $m \in \{m_1, m_2\}$ erbringen kann, muss er den anderen Beweis "simulieren". Der erste Beweis, genannt P_1, wird analog dem im Paragraph "Proof of Knowledge of Equality of Discrete Logarithm" beschriebenen Verfahren berechnet (da in diesem Beispiel c eine Verschlüsselung von m_1 ist). Das Commitment des zweiten Beweises, genannt P_2, wird hingegen nicht anhand von ω berechnet, sondern durch die zuerst zufällig gewählten Werte c_2 und s_2 (es gilt $t = g^s \cdot y^{-c}$).

Der Prover wählt zufällig den Wert $\omega_1 \in_R \mathbb{Z}_q$ und berechnet damit $t_{\alpha 1} = g^{\omega_1}$ und $t_{\beta 1} = h^{\omega_1}$ für den Teilbeweis P_1. Zur Simulation des Beweises von P_2 wählt er zufällig $c_2, s_2 \in_R \mathbb{Z}_q$. Die beiden Commitments zum Teilbeweis P_2 errechnet er danach wie folgt:

$$t_{\alpha 2} = g^{s_2} \cdot {y_{\alpha 2}}^{-c_2}, \quad t_{\beta 2} = h^{s_2} \cdot {y_{\beta 2}}^{-c_2}$$

Während in der interaktiven Variante an dieser Stelle der Austausch von c stattfinden würde, wird bei der nicht-interaktiven Variante wiederum ein Hashwert generiert, welcher die Commitments, sowie die y-Werte enthählt, für welche der Logarithmus bewiesen werden soll:

$$c = H(t_{\alpha 1}, t_{\beta 1}, y_{\alpha 1}, y_{\beta 1}, t_{\alpha 2}, t_{\beta 2}, y_{\alpha 2}, y_{\beta 2})$$

Die Challenge c_1 berechnet sich aus der Differenz zwischen c und c_2, also $c_1 = c - c_2$. Der Prover berechnet schliesslich $s_1 = \omega_1 + c_1 \cdot x_1$ und publiziert $(t_{\alpha 1}, t_{\beta 1}, t_{\alpha 2}, t_{\beta 2}, c_1, c_2, s_1, s_2)$.

Die Überprüfung des Beweises durch einen Verifier erfolgt in 3 Schritten:

1. überprüfen des Beweises P_1:

$$t_{\alpha 1} \cdot {y_{\alpha 1}}^{c_1} \stackrel{?}{=} g^{s_1} \tag{3.6}$$

$$t_{\beta 1} \cdot {y_{\beta 1}}^{c_1} \stackrel{?}{=} h^{s_1} \tag{3.7}$$

$$H(t_{\alpha 1}, t_{\beta 1}, y_{\alpha 1}, y_{\beta 1}) \stackrel{?}{=} c_1 \tag{3.8}$$

2. überprüfen des Beweises P_2:

$$t_{\alpha 2} \cdot {y_{\alpha 1}}^{c_2} \stackrel{?}{=} g^{s_2} \tag{3.9}$$

$$t_{\beta 2} \cdot {y_{\beta 1}}^{c_2} \stackrel{?}{=} h^{s_2} \tag{3.10}$$

$$H(t_{\alpha 2}, t_{\beta 2}, y_{\alpha 2}, y_{\beta 2}) \stackrel{?}{=} c_2 \tag{3.11}$$

3. überprüfen, ob die Summe der beiden Challenges c_1 und c_2 dem Hashwert der Commitments und y-Werte entsprechen:

$$H(t_{\alpha 1}, t_{\beta 1}, y_{\alpha 1}, y_{\beta 1}, t_{\alpha 2}, t_{\beta 2}, y_{\alpha 2}, y_{\beta 2}) \stackrel{?}{=} c_1 + c_2 \tag{3.12}$$

Die ersten beiden Schritte prüfen die miteinander verknüpften Beweise auf Ihre Gültigkeit. Mit dem dritten Schritt wird geprüft, ob alle Challenges korrekt sind und mittels verifizieren der Summe, dass alle Challenges enthalten sind. Implizit wird dadurch auch getestet, dass der Prover einen der Beweise (in unserem Fall P_1) nicht simulieren konnte, sondern selbst generieren musste.

3.3 Reencryption Mix-Networks

Bei ElGamal handelt es sich, wie bereits erwähnt, um ein multiplikativ homomorphes System. Durch diese Eigenschaft kann die Randomisierung eines ElGamal Tupels verändert werden, ohne dass der Aufwand zur Entschlüsselung steigt. Angenommen, beim ElGamal Tupel $c = (g^r, y^r \cdot m)$ soll die Randomisierung verändert werden (das Tupel soll wiederverschlüsselt werden). Dazu wird durch die Verschlüsselung der Nachricht $\tilde{m} = 1$ das ElGamal Tupel $\tilde{c} = (g^{r'}, y^{r'} \cdot 1)$ generiert, wobei r' die zufällig gewählte Randomisierung, y derselbe öffentliche Schlüssel wie aus c und 1 die Nachricht darstellt. Bei der Multiplikation von c mit $\tilde{c}$ entsteht nun $c' = (g^{r+r'}, y^{r+r'} \cdot m)$. Obwohl das Tupel wiederverschlüsselt wurde, kann durch Kenntnis von x (dem privaten Schlüssel zu y) die Nachricht m in einem Schritt entschlüsselt werden.

Ein *Mix-Network* wird zur Anonymisierung von Daten benutzt. Ein "Verifiable Reencryption Mix-Network" vereint nebst dem Durcheinandermischen (engl. shuffle) eine Wiederverschlüsselung in der Anwendung. Zusammen mit dem Resultat liefert es einen Beweis, dass weiterhin alle Einträge in unveränderter Form nach dem Mischen vorhanden sind, sowie keine neuen Einträge angefügt wurden.

Der Input eines Verifiable Reencryption Mix-Network ist eine Liste von Verschlüsselungen $c_1, \ldots, c_n$. Es ist π die Funktion zur Permutation der Elemente, welche die Menge der Verschlüsselungen entsprechend durcheinandermischt. Das Mix-Network permutiert nun die Liste und führt für jede Verschlüsselung eine

Wiederverschlüsselung $c_i \rightarrow c'_{\pi(i)}$ aus. Die Ausgabe ist somit eine Liste von Verschlüsselungen $c'_{\pi(1)}, \ldots, c'_{\pi(n)}$.

Die Ein- und Ausgabe eines Reencryption Mix-Networks ist in Abbildung 3.1 visualisiert.

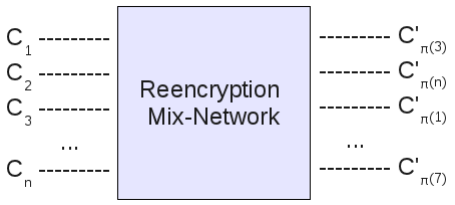


Abbildung 3.1: Input / Output eines Reencryption Mix-Network

Damit eine Anonymisierung der Daten stattfindet, sind mindestens zwei unabhängige Mix-Network Instanzen notwendig. Wird nur ein Mix-Network zur Anonymisierung verwendet, wäre diese Instanz in der Lage eine Verbindung zwischen den anonymisierten und den Ausgangsdaten, resp. zwischen Input und Output herzustellen. Um dieses Problem zu umgehen, werden mehrere Mix-Networks für die Durchführung der Anonymisierung eingesetzt.

3.4 Verteilte Entschlüsselung

Bei der Anwendung von asymmetrischer Kryptographie kann der Eigentümer des privaten Schlüssels ein Geheimnis alleine entschlüsseln (wenn die Nachricht an ihn verschlüsselt wurde). Alleine bedeutet, er ist nicht auf weitere Informationen oder Personen angewiesen. Im Kontext von E-Voting ist es wünschenswert, dass nicht eine Partei alleine, sondern erst der Verbund mehrerer Parteien in der Lage ist, eine Entschlüsselung durchzuführen. Die Aufteilung von privaten Schlüsseln bzw. Geheimnissen bezeichnen wir als *Secret Sharing*. Es sei hier angemerkt, dass die Generierung der Schlüssel ebenfalls ein Problem darstellt, da diese somit vor der Aufteilung mindestens einer Partei bekannt wären. Diese Problematik wird durch sogenannte *Distributed Key Generation*-Protokolle (DKG) gelöst.

Das einfache Secret Sharing Verfahren teilt ein Geheimnis s in $s_i, i \in \{1, \ldots, n\}$ Teile auf, wobei die Bedingung $s = s_1 + \ldots + s_n$ gilt. Alle s_i, $i > 1$ sind zufällig gewählt, das verbleibende Element s_1 stellt die Differenz von s und der Summe von $s_2, \ldots, s_n$ dar. Dadurch ist auch s_1 zufällig gewählt, wenn alle s_i,$i > 1$ zufällig gewählt wurden. Bei dieser Methode müssen alle Teilgeheimnisse bekannt sein, um die Rekonstruktion des Geheimnisses auszuführen (im Unterschied zu einem Threshold Verfahren).

Distributed Decryption: Die Aufteilung eines Geheimnisses dient dem Zweck, eine Entschlüsselung durch einen Verbund von Parteien durchzuführen (ohne dass eine einzelne Partei die gesamte Entschlüsselung vornehmen kann). Die Parteien errechnen mit Ihrem Teilschlüssel ein Zwischenresultat, welches von einer neutralen Instanz zuletzt "zusammengeführt" wird.

Jede Partei wählt zufällig einen Wert s_i, den Sie geheim hält. Weiter berechnet die Partei den öffentlichen Wert $y_i = g^{s_i}$ und veröffentlicht diesen zusammen mit einem NIZKP, welcher beweist, dass sie s_i zum Wert y_i kennt. Eine neutrale Instanz berechnet nach der Veröffentlichung aller y_i und Überprüfung aller NIZKPs den Gesamtschlüssel y mit

$$y = \prod_{i=1}^{n} y_i = g^{s_1 + \ldots + s_n}$$

Soll nun das ElGamal Tupel $c = (g^r, y^r \cdot m)$ verteilt entschlüsselt werden, wobei y der gemeinsame öffentliche Schlüssel darstellt, wird wie folgt verfahren:

Jede Partei P_i berechnet mit ihrem (Teil-)Geheimnis s_i den Wert $\omega_i = (g^r)^{s_i}$ und den NIZKP

$$ZKP\{(x) : \log y_i = x \wedge \log \omega_i = x\}$$

(g^r ist der erste Teil eines ElGamal Tupels, der Randomisierungsanteil). Nachdem alle ω_i berechnet sind, entschlüsselt die neutrale Instanz die Nachricht mit der Formel

$$m = \frac{(y^r \cdot m)}{\Omega}$$

, wobei Ω mit Hilfe der Gleichung

$$\Omega = \prod_{i=1}^{n} \omega_i = (g^r)^{s_1} \cdot \ldots \cdot (g^r)^{s_n} = (g^r)^{s_1+\ldots+s_n} = (g^r)^x$$

berechnet wird. Vor der Entschlüsselung werden die NIZKPs durch die neutrale Instanz verifiziert.

Durch diese Methode ist es möglich eine verteilte Entschlüsselung mit n Parteien vorzunehmen. Dieses Verfahren bedingt das korrekte Handeln aller Parteien. Dieser Umstand würde durch den Einsatz eines Threshold Verfahrens korrigiert, das die Schlüsselgenerierung sowie Entschlüsselung mit einem Teil der Parteien (Mindestanzahl) zulässt. Bei einem Threshold Verfahren kommt neben der Anzahl Parteien n der Threshold t hinzu, daher spricht man von einem n,t-Protokoll. Die verteilte Entschlüsselung kann als n,t-Protokoll mit $n = t$ betrachtet werden.

3.5 Plaintext Equivalence Test

Möchte man eine Aussage über zwei ElGamal Tupel bezüglich deren verschlüsselten Nachrichten treffen, kann man den "Plaintext Equivalence Test" (abgekürzt PET) durchführen. Dabei ist es wichtig zu beachten, dass bei einer solchen Aussage die Vertraulichkeit (engl. confidentiality) nicht verletzt wird. Im Kontext dieses Projekt liefert der PET eine Aussage darüber, ob zwei beliebige ElGamal Tupel Verschlüsselungen derselben Nachricht sind oder nicht, ohne jedoch die einzelnen Tupel zu entschlüsseln bzw. weitere Informationen über deren Inhalt aufzudecken. Dies steht im Unterschied zum Ansatz, ElGamal Tupel zu entschlüsseln und anschliessend zu vergleichen, womit die Geheimnisse offen gelegt würden.

Kennt man die beiden ElGamal Tupel $c_1 = (g^{r_1}, y^{r_1} \cdot m_1)$ und $c_2 = (g^{r_2}, y^{r_2} \cdot m_2)$ und will prüfen ob $m_1 \stackrel{?}{=} m_2$ gilt oder nicht (also ob c_1 und c_2 Verschlüsselungen derselben Nachricht sind), wendet man einen PET an. Dazu berechnet man das Tupel

$$\tilde{c} = \frac{(c_1)^z}{(c_2)^z} = \left(\frac{c_1}{c_2}\right)^z,$$

wobei z eine Zahl aus der Menge $\mathbb{Z}_q$ darstellt ($z \in \mathbb{Z}_q$). Durch die Entschlüsselung des Tupels $\tilde{c}$ erhält man eine Zahl, welche das Verhältnis der Werte ${m_1}^z$ und ${m_2}^z$ darstellt. Erhält man als Resultat der Entschlüsselung von $\tilde{c}$ den Wert "1", so ist ${m_1}^z = {m_2}^z$ und dadurch auch $m_1 = m_2$. Ist das Resultat ungleich "1", sind die beiden Nachrichten m_1 und m_2 unterschiedlich, ohne jedoch weitere Details zum Verhältnis der beiden Werte m_1 und m_2 bekannt gegeben zu haben. Der entscheidende Faktor ist die Entschlüsselung des ElGamal Tupels $\tilde{c}$ und nicht c_1 oder c_2.

Verteilter PET: Bei der Anwendung des verteilten PETs kommt es zu einer Entschlüsselung eines ElGamal Tupels, welche im bereits beschriebenen, verteilten Verfahren (verteile Entschlüsselung) durchgeführt wird. Die Generierung des Tupels $\tilde{c}$ bzw. dessen entschlüsselter Wert $\tilde{m}$ wird als "Verblindung" (engl. blinding) bezeichnet. Auch diese Verblindung wird durch mehrere Parteien gemeinsam durchgeführt. Dies führt zu folgendem Vorgehen jeder teilnehmenden Partei P_i:

1. zufällige Wahl von $z_i \in_R \mathbb{Z}_q$
2. Veröffentlichung des Wertes $Z_i = g^{z_i}$ und von $ZKP\{(z_i) : \log_g Z_i = z_i\}$

Diese Schritte werden von jeder Partei durchgeführt und durch die neutrale Instanz (und allenfalls weitere Parteien) kontrolliert. Aus der Addition der einzelnen z_i entsteht der gesamte Verblindungsfaktor z. Dieser darf dabei nicht so gewählt oder konstruiert sein, dass $z = 1$ gilt, da durch diesen Wert keine Verblindung stattfindet. Das "Commitment" der Parteien zu Ihrem Wert z_i stellt sicher, dass bei der späteren Berechnung der Teilresultate geprüft werden kann, ob derselbe Wert z_i verwendet wurde (und dadurch $z \neq 1$ gilt).

Es folgen die Schritte zur Berechnung der Teilresultate der Parteien für das Bestimmen des verteilten PETs für das ElGamal Tupel

$$\tilde{c} = \left(\frac{c_1}{c_2}\right)^z = \left(g^{r_1 - r_2}, y^{r_1 - r_2} \cdot m_1 \cdot m_2\right)^z = (A^z, B^z) :$$

1. Berechnung des Werts $\tilde{c}_i = \left(\frac{c_1}{c_2}\right)^{z_i} = (A^{z_i}, B^{z_i})$
2. veröffentlichen des Wertes $\tilde{c}_i$ mit dem Beweis $ZKP\{(z_i) : \log_g A^{z_i} = z_i \wedge \log_g B^{z_i} = z_i \wedge \log_g Z_i = z_i\}$

Die neutrale Instanz berechnet im Anschluss (nach Prüfung der Beweise) das gesamte ElGamal Tupel $\tilde{c}$ mit Hilfe der Gleichung

$$\tilde{c} = \prod_{i=1}^{n} \tilde{c}_i = \left(\frac{c_1}{c_2}\right)^{z_1 + \ldots + z_n} = \left(\frac{c_1}{c_2}\right)^z .$$

Das Tupel $\tilde{c}$ wird nun, wie bereits beschrieben, verteilt entschlüsselt und das Resultat der Entschlüsselung interpretiert.

4 Aufbau der Applikation

Die Applikation ist in mehrere Teile aufgetrennt, welche den entsprechenden Rollen innerhalb des Protokolls nachempfunden sind. Jeder Applikationsteil ist selbständig ausführbar und in einem realistischen Szenario auf einem separaten Computer im Einsatz. Die Applikationsbereiche werden nachfolgend aufgelistet, wobei die mit einem * gekennzeichneten Bereiche mehrfach vorkommen (von diesen Teilen gibt es mehrere Instanzen):

- Bulletin Board
- *Voter GUI (Voter Client)
- Admin GUI (Admin Client)
- Registrar Authority
- *Tallying Authority
- Board Authority
- *Mixing Authority
- Verifier

Bevor die einzelnen Bereiche genauer beschrieben werden, soll kurz die Funktion dieser Applikationsteile aufgezeigt werden.

Bulletin Board: Das Bulletin Board dient als öffentlich zugänglicher "Speicher" für die verschiedenen Einträge, welche im gesamten Prozess entstehen. Damit der korrekte Ablauf sichergestellt werden kann, befindet sich das Board während des gesamten Prozesses in verschiedenen Zuständen (vgl. Abschnitt 4.3). Diese Zustände legen fest, welche Aktionen ausgeführt werden dürfen.

Voter GUI: Dieser Teil stellt die Applikation für einen Wähler dar. Mithilfe der Voter GUI Applikation kann der Wähler seine Stimme auf dem Board ablegen.

Admin GUI: Das Admin GUI wird benutzt, um die verschiedenen Stati des Bulletin Boards zu kontrollieren bzw. die Statuswechsel vorzunehmen. Weiter werden mit Hilfe des Admin GUIs der Abstimmungstext und die Wahloptionen festgelegt. Während des Abstimmungsprozesses kann über das Admin GUI die Anzahl der verschiedenen Objekte auf dem Bulletin Board verfolgt werden.

Registrar Authority: Die Aufgabe der Registrar Authority ist der Rolle "Registrar" gemäss dem Protokoll JCJ05 nachempfunden. Ihre Aufgabe ist es, die Wählerregistrierung vorzunehmen bzw. eine Methode zur Registrierung eines Wählers anzubieten und die Wählerliste (Voter Roll) im Status "Setup" des Bulletin Boards auf dem Board abzulegen.

Tallying Authority: Dieser Applikationsteil implementiert die Aufgaben einer Tallying Authority. Diese sind gemäss dem Protokoll das verteilte Generieren eines öffentlichen Schlüssels (der auf dem Bulletin Board abgelegt wird), das verteilte Verblinden und das verteilte Entschlüsseln von ElGamal Tupeln. Die Tallying Authorities arbeiten dabei einerseits mit dem Bulletin Board (ablegen Ihrer Teilresultate) sowie mit der Board Authority (informieren über den Abschluss ihrer Arbeit) zusammen.

Board Authority: Die Board Authority ist zuständig dafür, dass aus den Teilresultaten der einzelnen Tallying Authorities das Gesamtrestultat berechnet wird. Weiter führt die Board Authority gewisse Kontrollen auf den abgegebenen Stimmen aus. Konkret werden die Phasen "Invalid Vote Elimination" und "Ballot Replication" durch die Board Authority ausgeführt.

Mixing Authority: Der Applikationsbereich Mixing Authority ist eine vereinfachte Implementation eines "Reencryption Mix-Networks" (vgl. Abschnitt 3.3). Die Aufgabe dieser Instanzen besteht im Vermischen und Wiederverschlüsseln der abgegebenen Stimmen.

Verifier: Dieser Applikationsteil repräsentiert einen Benutzer (der nicht zwingend auch für die Abstimmung zugelassen ist), welcher die verschiedenen Berechnungen überprüft. Die Prüfung erfolgt dabei durch erneutes Nachrechnen der verschiedenen Schritte.

4.1 Eingesetzte Fremdkomponenten

Die nachfolgenden Komponenten wurden bei der Erstellung der Applikation verwendet. Zu jeder Komponente wird kurz der Verwendungszweck während dieser Arbeit beschrieben.

Eclipselink: Dieses Framework wurde für die Implementation der Java Persistence API (JPA) zur Speicherung der Java Objekte in eine Datenbank eingesetzt.
`http://www.eclipse.org/eclipselink`

MySQL: Als Datenbank wurde während dieser Arbeit eine MySQL Instanz eingesetzt. Aufgrund der eingesetzten JPA Architektur kann die Datenbank aber relativ einfach ausgetauscht werden (vgl. Abschnitt 5.5).
`http://www.mysql.com`

Apache ANT: Zur Unterstützung des Deployment Prozesses sowie zur Erstellung der Javadoc wurde das Apache ANT Buildsystem benutzt.
`http://ant.apache.org`

Apache log4j: Für die Umsetzung des Logging-Prozesses der Applikationen wurde das log4j Logging Framework "log4j" der Apache Software Foundation eingesetzt.
`http://logging.apache.org/log4j`

Apache Commons cli: Die Apache "Commons cli" Bibliothek bietet ein Set von Methoden / Operationen zur Verarbeitung von Kommandozeilen-Parametern an.
`http://commons.apache.org/cli`

Jigloo: Für die Erstellung der beiden GUIs (Wähler und Administrator) wurde CloudGarden's Jigloo SWT / Swing GUI Builder eingesetzt.
`http://www.cloudgarden.com`

Metrics: Eclipse Plug-In für die Ermittlung von statistischen Daten zum Java Projekt wie z.B. die Anzahl Codezeilen und Methoden.
`http://metrics.sourceforge.net`

4.2 Architektur

In den nun folgenden Abschnitten werden die einzelnen Applikationsbereiche im Detail beschrieben. Dies umfasst neben der Struktur auch Informationen über die Implementation sowie mögliche Einschränkungen. Abhängig von der Komplexität des Applikationsteils werden die einzelnen Aufgaben dieses Bereiches zusätzlich erläutert.

Abbildung 4.1 zeigt die verschiedenen Applikationsbereiche, sowie deren Interaktion untereinander. Die Kommunikation unter den verschiedenen Applikationsteilen wird durch das Java RMI Framework realisiert. RMI steht für Remote Method Invocation und ermöglicht verteilte Anwendungen. Die Pfeile deuten jeweils die Nutzung der RMI-Server Objekte durch die verschiedenen Applikationsteile an. Die Parteien "Registrar Authority" und "Board Authority" sind speziell, da sie RMI-Services anbieten aber auch als RMI-Client die Services des Bulletin Boards nutzen. Diese beiden Applikationsteile sind daher sowohl RMI-Server als auch

RMI-Client. Die Applikationsteile Bulletin Board und Registrar Authority verwenden zudem zur Speicherung der Daten eine Datenbank. Die Datenbankanbindung wurde, wie bereits erwähnt, mit dem Eclipselink Framework realisiert und kann über entsprechende Konfigurationsdateien konfiguriert werden (siehe Abschnitt 5.5).

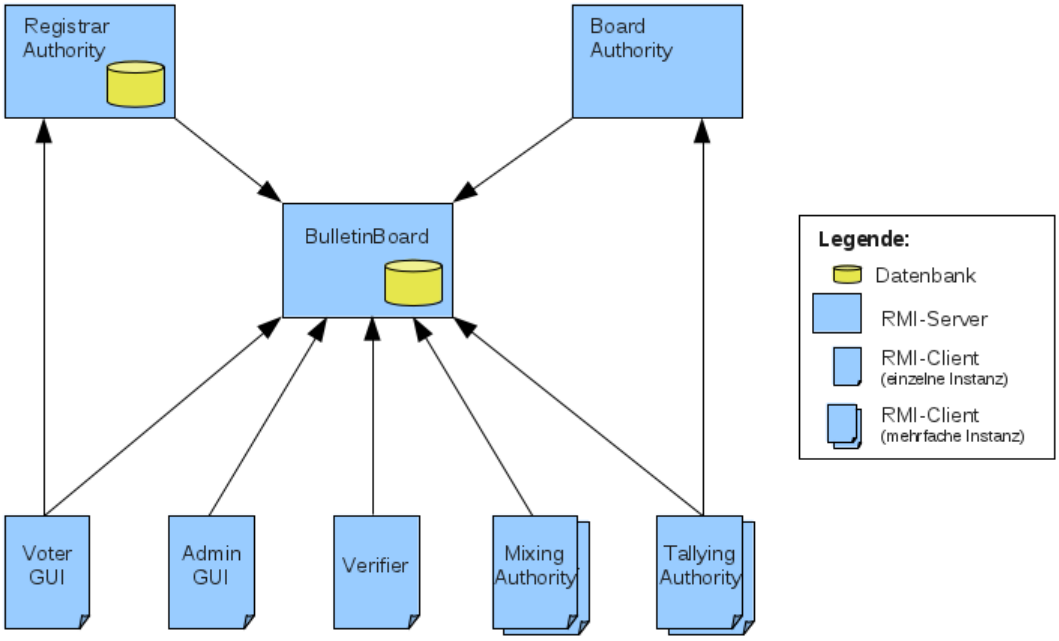


Abbildung 4.1: Applikationsbereiche

4.3 Bulletin Board

Als Bulletin Board bezeichnet man den Teil einer E-Voting-Applikation, welcher für die Aufnahme und Speicherung der Stimmen zuständig ist. Das Bulletin Board stellt die zentrale Einheit der entwickelten Applikation dar, auf ihm legen die Wähler ihre Stimmen ab. Die Umsetzung des Protokolls generiert zudem weitere Daten, welche ebenfalls auf dem Bulletin Board publiziert werden. Zu diesen Daten zählen:

- die Wählerliste (Voter Roll)
- die Wahloptionen (Candidate Slate)
- Einträge über ungültige, doppelte oder gefälschte Stimmen (Invalid, Duplicate oder Fake Votes)
- Teilresultate der Berechnungen der Tallying Authorities
- Resultate durch den Mixing-Prozess (Resultate der Mixing Authorities)
- Gesamtresultat der Abstimmung

Das Bulletin Board befindet sich während dem gesamten Abstimmungsprozess in einem festgelegten Zustand, welcher durch einen Administrator (mit Hilfe des Admin GUIs) gesetzt wird. Das Zustandsdiagramm in Abbildung 4.2 zeigt den Ablauf und die auftretenden Zustände des Bulletin Boards. Die Zustände werden nachfolgend einzeln beschrieben.

Zustand "Setup": Die Abstimmung wird eingerichtet. Der Ablauf für die Einrichtung ist definiert im Use Case A.3. Nach Abschluss der Einrichtung wird die Abstimmung freigegeben und das Board in den Zustand "Vote Casting" überführt.

Zustand "Vote Casting": In diesem Zustand erfolgt die Stimmabgabe durch die Wähler und das Publizieren der Stimmen auf dem Board. Der Ablauf dieser Aktion ist im Use Case A.1 beschrieben.

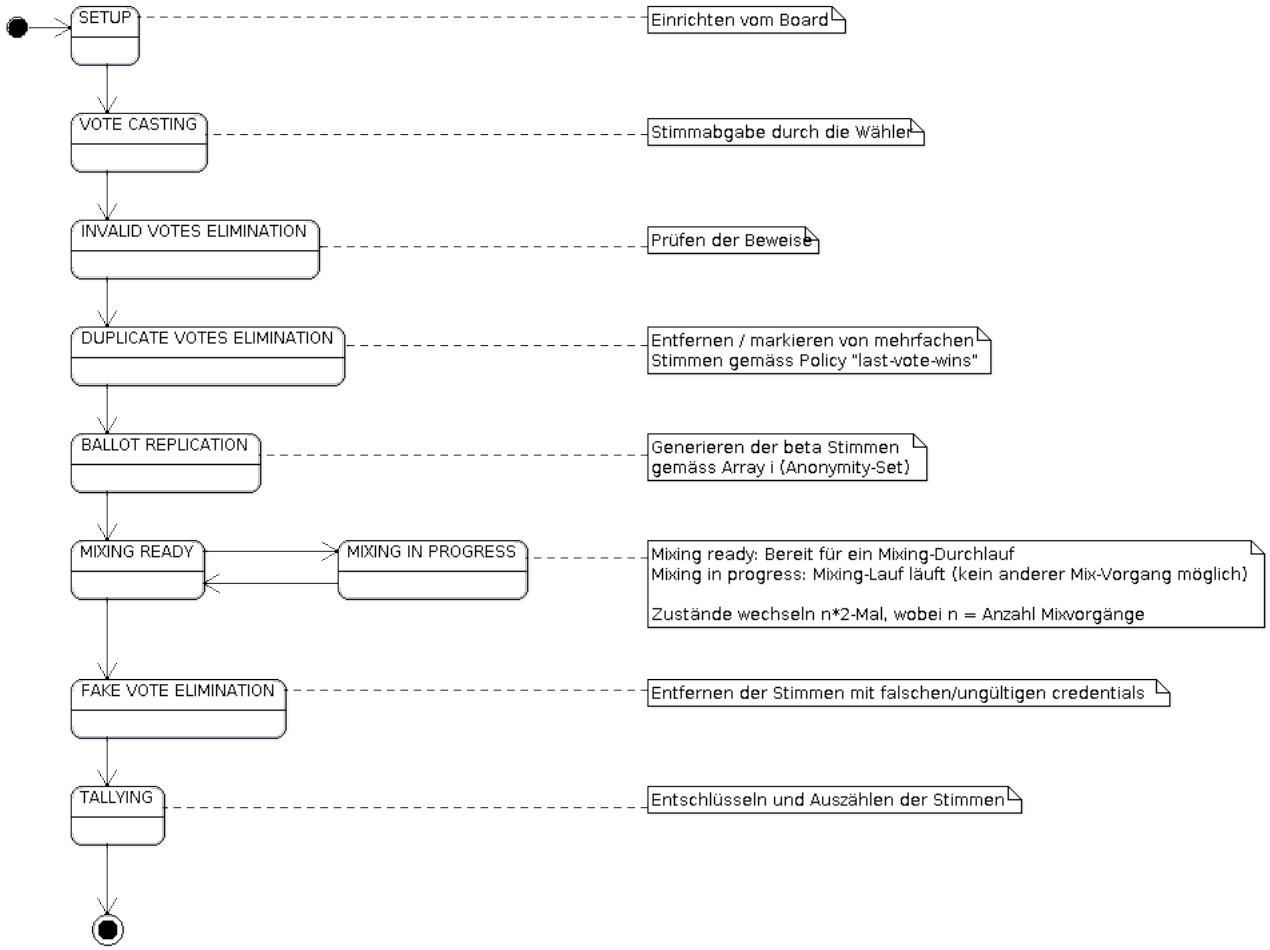


Abbildung 4.2: Zustandsdiagramm Bulletin Board

Zustand "Invalid Vote Elimination": Am Anfang der "Vote Authorization" steht das Markieren derjenigen Stimmen (Einträge vom Typ BallotEntry), für welche die NIZKPs nicht korrekt sind (oder fehlen). Die Markierung wird in Form eines neuen Eintrags vom Typ InvalidVoteEntry (mit Verweis auf die nicht korrekte Stimme) umgesetzt. Die möglichen Typen von Einträgen auf dem Bulletin Board werden im Abschnitt 4.3.1 beschrieben.

Zustand "Duplicate Vote Elimination": In diesem Zustand werden Stimmen markiert, welche vom selben Wähler stammen. Welche Stimme letztendlich gezählt werden soll, muss vorgängig definiert werden. In der implementierten Variante zählt bei einer mehrfachen Stimmabgabe jeweils die letzte, gültige Stimme ("last-vote-wins").

Zustand "Ballot Replication": Nachdem die ungültigen und mehrfachen Stimmen markiert wurden, werden in der Phase "Ballot Replication" (wie im Abschnitt 2.2 beschrieben) durch die neutrale Instanz aus einer abgegebenen Stimme der Form A, B, I insgesamt i Stimmen der Form A, B, S_i generiert. A ist die verschlüsselte Wähler-Identität, B die verschlüsselte Wahl, I eine Liste von Wählerindizes und S_i die von der Registrar Authority verschlüsselte Wähler-Identität aus der Voter Roll (Index i). In diesem Schritt werden aus einer Stimmabgabe also insgesamt i Stimmen generiert, entsprechend der Anzahl des Parameters "BETA" bzw. "BETAMAX" des Bulletin Boards (vgl. Abschnitt 5.5).

Zustände "Mixing Ready" und "Mixing In Progress": Die Stimmen werden nun anonymisiert, damit keine Rückschlüsse auf die ursprünglichen Wähler möglich sind. Sobald die Liste der nicht markierten Stimmen (Einträge vom Typ "BallotEntry", für welche weder ein Eintrag vom Typ "InvalidVoteEntry" noch vom Typ "DuplicateVoteEntry" existiert) von einer Mixing Authority gelesen wird, wechselt der Zustand von "Mixing Ready" auf "Mixing In Progress". Die einzelnen Mixing-Durchläufe müssen sequentiell ablaufen. Eine Mixing Authority führt für jedes Tupel eine Wiederverschlüsselung durch und ordnet die Reihenfolge der Liste neu (mischen, engl. mixing).

Sobald der Output der Mixing Authority in Form von "Mixed Ballot Entries" zurück auf das Bulletin Board geschrieben wurde, wechselt der Zustand zurück auf "Mixing Ready" (automatisiert durch das Bulletin Board, ohne Eingriff des Administrators) und die nächste Mixing Authority kann ihre Arbeit aufnehmen. Bei n Mixing Durchläufen finden folglich insgesamt $2n$ Wechsel von diesen beiden Zuständen statt. Nach Durchführung des letzten Mixing Durchlaufs wird eine entsprechende Variable gesetzt, welche bestätigt, dass die Mixing Phase abgeschlossen wurde.

Zustand "Fake Vote Elimination": In diesem Zustand wird die Liste der Stimmen aus dem letzten Mixing Output durchlaufen und es wird getestet, ob die ElGamal Tupel A und S der jeweiligen Stimme dieselbe Wähleridentifikation darstellen. Dazu wird der im Abschnitt 3.5 beschriebene verteilte PET angewendet. Alle Einträge, für welche $PET(A, S)$ "false" zurück liefert, werden durch einen Fake Vote Entry markiert.

Zustand "Tallying": Im letzten Schritt wird wiederum die Liste der Stimmen aus dem letzten Mixing Output gelesen abzüglich jener Stimmen, für welche eine FakeVoteEntry-Markierung existiert. Die Stimmen werden im Verbund entschlüsselt und gezählt. Konkret wird ein Zähler für die entsprechende Wahloption inkrementiert.

Das Bulletin Board bietet für die Nutzer (Wähler, Wahlbehörde, Mixing Authority, Tallying Authority, Board Authority, Verifier und Administrator) verschiedene Operationen an. Das Board stellt dabei sicher, dass die angebotenen Operationen nur im entsprechenden Zustand ausgeführt werden. So ist es beispielsweise nur während dem Zustand "Vote Casting" möglich, als Wähler eine Stimme auf dem Board abzulegen. Die Beschreibung der Operationen des Bulletin Boards folgt im Abschnitt 4.11.1.

4.3.1 Board Einträge

Wie in den Projektzielen im Abschnitt 1.1.4 definiert, ist ein Bulletin Board Eintrag grundsätzlich sehr generisch. Jeder Eintrag auf dem Bulletin Board ist ein Objekt einer Unterklasse vom Typ "BoardEntry" und enthält eine Objekt ID (Attribut "ID"), einen Zeitstempel ("timestamp") sowie den Board-Zustand, in welchem das Objekt generiert wurde ("state"). In Abbildung 4.3 sind die verschiedenen Einträge, welche auf dem Board abgelegt ersichtlich. Die verschiedenen Einträge werden nachfolgend einzeln beschrieben.

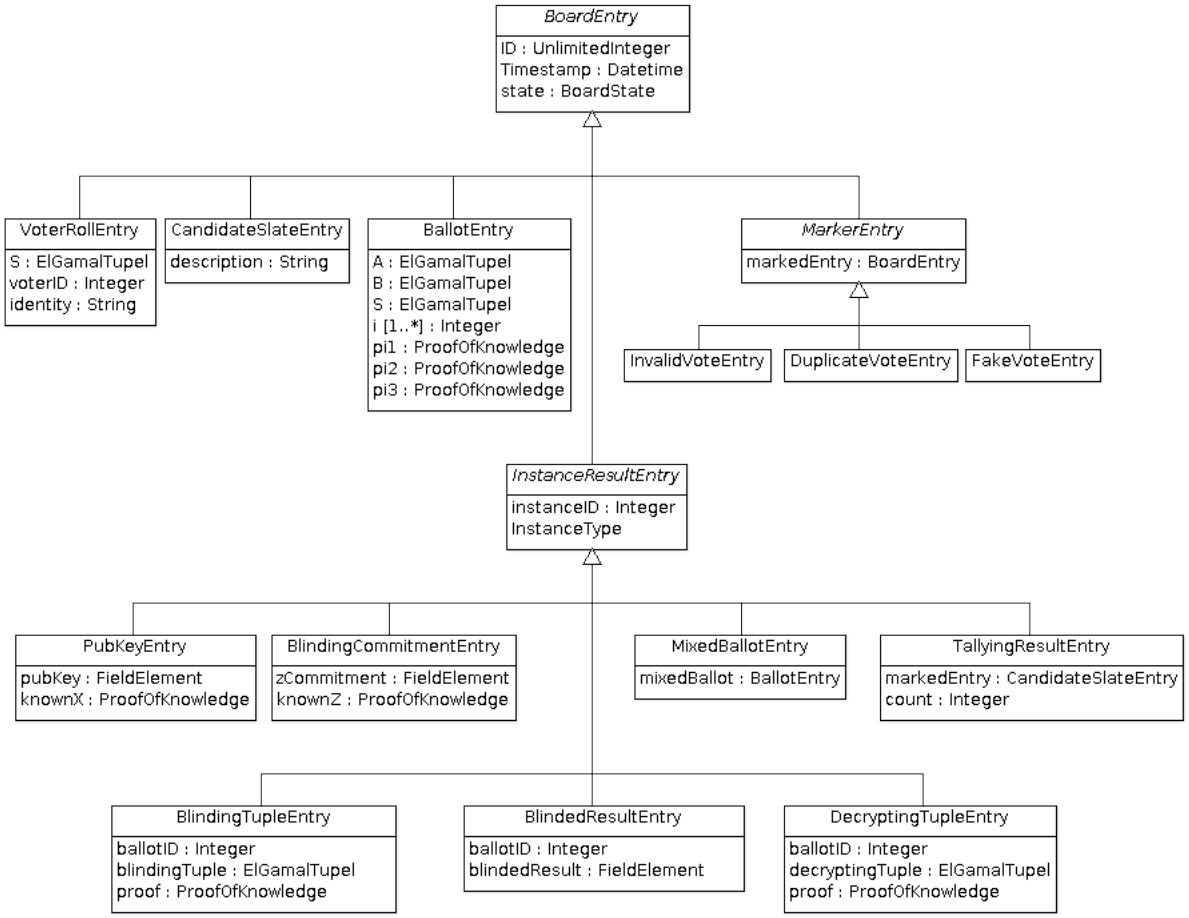


Abbildung 4.3: Klassendiagram BoardEntry

Voter Roll Entry: Stellt einen Eintrag der vom Registrar publizierten Voter Roll (Wahlregister) dar. Das Objekt beinhaltet Attribute für das verschlüsselte Credential in Form eines ElGamal Tupels, die Bezeichnung des Wählers sowie seine Voter ID (zur Identifikation aus der Voter Roll).

Candidate Slate Entry: Ein Objekt vom Typ "CandidateSlateEntry" spezifiziert eine Wahloption, welche durch den Wahladministrator festgesetzt wurde und dem Wähler bei der Stimmabgabe zur Auswahl steht. Einträge dieses Typs erweitern das Objekt "BoardEntry" um die Bezeichnung der Option ("description").

Ballot Entry: Jede Stimmabgabe resultiert auf dem Board in einem Eintrag vom Typ "BallotEntry". Gemäss Protokoll besteht ein Ballot Entry aus:

- zwei ElGamal Verschlüsselungen (Voter Credential A, gewählte Wahloption B),
- dem vom Registrar verschlüsselten Credential des Wählers S bzw. dem Anonymitätsset I (eine Liste von Indizes aus der VoterRoll),
- dem Proof of Knowledge of Plaintext or Randomness für die Kenntnis des Voter Credentials,
- dem Proof of Knowledge of Plaintext or Randomness für die Kenntnis der gewählten Wahloption,
- dem Proof of Validity, um zu beweisen, dass eine gültige Wahloption gewählt wurde.

Abhängig vom Zustand, in welchem die Stimme auf dem Bulletin Board gespeichert wird ("Vote Casting", "Ballot Replication" oder "Mixing In Progress"), enthält ein Eintrag entweder die Liste der referenzierten Voter Roll IDs (das Anonymitätsset) oder die von der Registrar Authority verschlüsselte Identität des Wählers (S).

Marker Entry: Das Protokoll sieht vor, dass in gewissen Stati abgegebene Stimmen markiert werden. Dabei treten die Markierungen **InvalidVote**, **DuplicateVote** und **FakeVote** auf. Diese Einträge erweitern die Klasse "MarkerEntry" (der markierte Ballot Entry wird mit dem Attribut "markedBallot" referenziert).

Instance Result Entry: Neben den Marker Entries gibt es (Teil)Resultate, welche auf dem Board gespeichert werden. Zu diesen Resultaten zählen etwa die Teil-Öffentlichen Schlüssel jeder Tallying Authority oder das Teil-Resultat der Entschlüsselung einer Stimme. Alle diese Einträge erweitern die Klasse "InstanceResultEntry". Einträge dieses Typs besitzen Attribute für die Instance ID (Identifikation der Instanz, welche das Resultat berechnet hat) wie auch des Instanz Typs. Es treten folgende Instanz-Typen auf:

- Tallying Authority
- Mixing Authority
- Board Authority

Die nachfolgende Liste beschreibt die verschiedenen Teilresultate, welche auf dem Board gespeichert werden:

- **PubKey Entry**: Der öffentliche Teilschlüssel einer einzelnen Tallying Authority zusammen mit dem Beweis der Kenntnis des diskreten Logarithmus (Kenntnis vom zugehörigen privaten Schlüssel).
- **Blinding Commitment Entry**: Das Commitment zum gewählten "Blinding"-Faktor einer Tallying Authority. Jede Instanz wählt wie beim Verfahren "Distributed PET" (vgl. Abschnitt 3.5) beschrieben einen Verblindungsfaktor z und berechnet $Z = g^z$ als Commitment dazu. Das Commitment wird zusammen mit dem Beweis der Kenntnis von z (Proof of Knowledge of Discrete Logarithm) auf dem Board abgelegt.
- **Blinding Tuple Entry**: Das Teilresultat der Verblindung durch eine Tallying Authority. Jede Instanz muss neben dem Resultat auch den Beweis über die Verwendung des Verblindungsfaktors z, zu welchem sie sich verpflichtet hat, mitliefern. Der Eintrag speichert zudem die Ballot ID des Ballot Entries, für welchen die Verblindung stattfindet. Einträge dieses Typs werden in den Phasen "Duplicate Vote Elimination" und "Fake Vote Eliminiation" verwendet (Verblindung des Voter Credentials und $PET(A, S)$).

- **Blinded Tuple Entry**: Nachdem alle Tallying Authorities ihre Teilresultate bezüglich der Verblindung abgelegt haben, berechnet die Board Authority das Gesamtresultat und speichert dieses in Form von BlindedTupleEntry-Objekten auf dem Board. Die Einträge "Blinding Tuple Entry" und "Blinded Tuple Entry" sind nicht zu verwechseln: Blinding bedeutet ein Teilresultat (berechnet durch eine Tallying Authority), Blinded bedeutet das Gesamtresultat (errechnet durch die Board Authority). Im Unterschied zu einem Blinding Tuple Entry wird bei einem Blinded Tuple Entry kein Beweis mehr gespeichert. Die Korrektheit der Berechnung der Board Authority wird durch den Applikationsteil "Verifier" vollzogen.
- **Decrypting Tuple Entry**: Dieser Eintrag enthählt das Teilresultat einer Tallying Authority für die Entschlüsselung eines ElGamal Tupels. Gespeichert wird jeweils die Ballot ID des Ballot Entries, welcher entschlüsselt wird, das Resultat der Berechnung (als ElGamal Tupel) und der Beweis für die Verwendung des korrekten privaten Schlüssels. Einträge dieses Typs treten in den Phasen "Duplicate Vote Elimination", "Fake Vote Elimination" und "Tallying" auf.
- **Mixed Ballot Entry**: Dieser Typ stellt ein Ballot Entry nach dem Mixing durch eine Mixing Authority dar. Jede Mixing Authority liefert als Resultat des Shuffle- und Reencryption-Prozesses eine Liste von Stimmen, welche als Mixed Ballot Entries auf dem Board abgelegt werden.

4.4 Voter Client / Voter GUI

Die Funktionalität des Wählers wurde in Form der Klasse "VoterClient" realisiert. Ein Voter Client wird entweder vom Wähler selbst über das Voter GUI oder durch durch die Simulation für die simulierte Stimmabgabe instanziert. Folgend der Ablauf einer Stimmabgabe innerhalb der Klasse "VoterClient":

1. Zufälliges Wählen einer Randomisierung mit Hilfe des im Abschnitt 4.12.3 erwähnten SecureRandom-Generators.
2. Erzeugen der ElGamal Tupel A (Identität) und B (Wahl) mittels der Klasse ElGamalSystem (siehe Abschnitt 4.12.1).
3. Wahlregister laden und über die Methode java.util.Collections.shuffle() (siehe Anhang C) permutieren.
4. Zufälliges Ermitteln der Grösse des Anonymitätssets, abhängig von den beiden Parametern "BETA" und "BETAMAX" (vgl. Abschnitt 5.5).
5. Hinzufügen der eigenen Identität und weiteren Identitäten aus dem permutierten Wahlregister bis die Grösse des Anonymitätssets erreicht wurde. Da die verwendete Datenstruktur HashSet ungeordnet ist, existiert keine Reihenfolge. Die zuerst eingefügte, legitime Identität kann nach dem Hinzufügen der "falschen" Identitäten an einer beliebigen Stelle innerhalb des Anonymitätssets stehen. Das HashSet verhindert zudem implizit Duplikate. Sollte zufälligerweise die (bereits verwendete) legitime Identität zum HashSet hinzugefügt werden, so wird diese einfach überschrieben. Das HashSet garantiert folglich die Anzahl Identitäten nach Vorgabe des Anonymitätssets ohne Duplikate.
6. Generieren der Beweise π_1 (Identität), π_2 (Wahl) und π_3 (gültige Wahl aus Candidate Slate). Bei der Generierung der drei Beweise wird dieselbe Challenge (Hashwert über die relevanten Informationen aller drei Beweise) verwendet, damit eine Verknüpfung entsteht. Andernfalls wäre es möglich, einzelne Beweise zu kopieren (z.B. den Beweis der Kenntnis der Wahl).
7. Der Rückgabewert der Methode ist schliesslich ein Objekt vom Typ "BallotEntry", welches die ElGamal Tupel A und B, das Anonymitätsset sowie die drei generierten Beweise enthält.

Der generierte Ballot Entry wird anschliessend durch das Voter GUI, bzw. durch die Simulation auf dem Board abgelegt. Die Benutzung des Voter GUIs ist im Abschnitt 5.4 beschrieben.

4.5 Admin Client / Admin GUI

Die Funktionalität des Admin Clients ist direkt in die Java Klasse "AdminGUI" integriert. Diese Klasse instanziert ein neues Objekt vom Typ "BoardServiceAdmin", welches als Schnittstelle zum Bulletin Board verwendet wird. Folgende Werte werden beim Starten des Admin GUIs automatisch vom Bulletin Board abgefragt (sofern vorhanden) und angezeigt:

- Wahlregister (Voter Roll)
- Wahloptionen (Candidate Slate)
- Abstimmungstext
- Boardstatus

Beim Speichern der Setup Parameter werden der Abstimmungstext und die Wahloptionen auf das Bulletin Board geschrieben. Im Falle des Candidate Slates geschieht dies in Form von Candidate Slate Entries. Da die Applikation gemäss Projektziele (Abschnitt 1.1) kein Löschen von Einträgen erlauben soll, ist das Setzen des Candidate Slates nur ein Mal erlaubt. Existieren bereits Candidate Slate Einträge auf dem Board, wird eine entsprechende Exception geworfen. Exceptions werden, wie im Abschnitt 4.13.4 spezifiziert, an das Admin GUI weiter gereicht und dort - sofern sinnvoll - dem Administrator in Form einer Meldung angezeigt. Der aktuelle Status der Abstimmung kann vom Wahladministrator über die Schaltfläche “Aktualisieren” abgefragt werden.

Die Verwendung des Admin GUIs aus Sicht des Administrators wird im Abschnitt 5.3 beschrieben.

4.6 Registrar Authority

Grundsätzlich übernimmt die Registrar Authority die Aufgaben der Wähler Registrierung sowie der Publizierung des Wahlregisters. Der erste Schritt wird gemäss Protokoll durch den Verbund mehrerer Registrar Authorities umgesetzt. Der Applikationsteil der Registrar Authority wurde sehr rudimentär umgesetzt, wie dies zwischen den Betreuern und den Studenten vereinbart wurde (vgl. Abschnitt 1.1.4). An dieser Stelle soll daher kurz erwähnt werden, wie die Registrierung korrekterweise ablaufen sollte und wie diese im Rahmen eines Folgeprojekts bzw. einer Weiterentwicklung implementiert werden könnte.

Der Use Case A.4 zeigt den Ablauf der Wählerregistrierung bzw. die auftretenden Schwierigkeiten. Um diesen Ablauf sicher umsetzen zu können, wäre eine Applikation zu entwickeln, welche unter anderem auch eine beidseitige Authentifikation (engl. mutual authentication) durchführt. Weiter sollte das für die Abstimmung benutzte Voter Credential nicht einer einzelnen Registrar Authority bekannt sein. Beide Eigenschaften fehlen in der Implementation der Registrar Authority, die in dieser Arbeit enthalten ist. Da jedoch ohne Registrar Authority keine Abstimmung umsetzbar ist, wurde eine Art “dummy” Registrar Authority umgesetzt. Die Funktionen der in dieser Arbeit verwendeten Registrar Authority sind:

- speichern der Liste der Wähler (in einer eigenen Datenbank)
- generieren und publizieren der Liste der wahlberechtigten Personen (Voter Roll)

Während bei einer realen, verteilten Registrar Authority die Voter Credentials nur verschlüsselt in Form von ElGamal Tupeln vorliegen bzw. gespeichert würden, sind die Credentials in der entwickelten Implementation unverschlüsselt in der Datenbank der Registrar Authority gespeichert. Der Grund für diese Umsetzung liegt bei der Simulation: Beim Starten der Registrar Authority ist es möglich, über die Konfigurationsdatei (siehe Abschnitt 5.5) festzulegen, ob eine Methode zur simulierten Stimmabgabe der gespeicherten Voter Credentials angeboten wird oder nicht. Die simulierte Stimmabgabe bedeutet, dass für jedes gespeicherte Voter Credential durch die Registrar Authority ein Aufruf des Voter Clients gemacht wird. Dies ist nur möglich, weil die Implementation der Registrar Authority in dieser Arbeit die Voter Credentials “kennt”, was in einer realen Umgebung selbstverständlich nicht der Fall wäre.

Weitere Gründe für die gewählte Umsetzungsvariante sind im Abschnitt 6.2 beschrieben.

4.7 Tallying Authority

Eine Tallying Authority ist eine Instanz, welche Berechnungen zu einzelnen Phasen der Abstimmung durchführt und ein entsprechendes Teilresultat liefert. In dieser Arbeit wird, wie bereits erwähnt, das sogenannte “distributed” Verfahren eingesetzt, daher müssen sich zwingend alle Tallying Authority Instanzen beteiligen. Die Teilresultate werden zusammen mit der Identifikation der jeweiligen Tallying Authority auf dem Bulletin Board abgelegt und durch die Board Authority zum Gesamtresultat zusammengerechnet.

Eine Tallying Authority berechnet im gesamten Prozess der Abstimmung die nachfolgenden Elemente:

Berechnung des "Public Keys": Im Board Status "Setup" berechnet jede Tallying Authority i ihren öffentlichen Schlüssel $y_i = g^{x_i}$ mit Hilfe des zufällig gewählten Wertes $x_i \in_R \mathbb{Z}_q$. Mit diesem Wert berechnet die Tallying Authority einen "Proof of Knowledge of Discrete Logarithm" (vgl. Abschnitt 3.2) um zu beweisen, dass sie den diskreten Logarithmus x_i zum Wert y_i (den privaten Teil des Schlüsselpaars) kennt. Aus diesen beiden Werten entsteht ein Objekt der Klasse "PubKeyEntry" (vgl. Abschnitt 4.3.1).

Berechnung des "Blinding Factors": Ebenfalls während der Phase "Setup" des Bulletin Boards berechnet jede Tallying Authority i einen sogenannten "Blinding Factor" $z_i \in_R \mathbb{Z}_q$. Dieser Faktor wird benutzt, um die verschlüsselten "Voter Credentials" aus den abgegebenen Stimmen (Objekte vom Typ "BallotEntry") während den Phasen "Duplicate Vote Elimination" und "Fake Vote Elimination" zu verblinden (vgl. auch Abschnitt 3.5). Damit die Tallying Authority denselben Faktor verwendet (und so auch keine Instanz durch geschickte Wahl eines Faktors eine Verblindung auflösen bzw. verhindern kann), muss sie sich zum gewählten Faktor verpflichten (ein "Commitment" zum Faktor z_i abgeben). Das Commitment Z_i berechnet sich aus $Z_i = g^{z_i}$ und wird zusammen mit dem Beweis über die Kenntnis von z_i auf dem Bulletin Board als "Blinding Commitment Entry" abgelegt.

Teilverblinden der "Voter Credentials": In der Phase "Duplicate Vote Elimination" berechnen alle Tallying Authorities anhand ihres Verblindungsfaktors z eine Teilverblindung, welche sie in Form von "BlindingTupleEntry" Objekten auf dem Bulletin Board ablegen. Im Falle der Verblindung der "Voter Credentials" berechnet jede Tallying Authority i für jede abgegebene Stimme (Ballot Entry) auf dem ElGamal Tupel A (das verschlüsselte "Voter Credential") die Verblindung mit $A^{z_i} = (g^{r\,z_i}, (g^r \cdot m)^{z_i}) = (a, b)$ und legt das Resultat der Berechnung zusammen mit dem Beweis

$$ZKP\{(z_i) : \log_{g^r} a = \log_{y^r} b = \log_g Z_i\}$$

auf dem Bulletin Board ab. Der mitgelieferte Beweis stellt sicher, dass die Tallying Authority für beide Teile des ElGamal Tupels denselben "Blinding Factor" verwendet hat und dies derselbe Faktor ist, zu dem sie sich in der Phase "Setup" verpflichtet hat.

Teilentschlüsseln der verblindeten "Voter Credentials": Nachdem alle Tallying Authorities gemeinsam die Voter Credentials verblindet haben, werden diese entschlüsselt. Dies geschieht ebenfalls im Verbund: Jede Tallying Authority i berechnet ihr Teilresultat durch Potenzieren des Randomisierungsanteils a aus dem zu entschlüsselnden ElGamal Tupel c mit dem Wert ihres privates Schlüssels x_i:

$$\tilde{A}_i = (a, b) = (a^{x_i}, b) = (g^{r\,x_i}, y^r \cdot m)$$

Das Tupel $\tilde{A}_i$ stellt das durch die Tallying Authority i teilentschlüsselte Tupel A dar. Diese Teilresultate werden zusammen mit dem Beweis der Gleichheit des diskreten Logarithmus vom öffentlichen Schlüssel y_i zur Basis g und dem Randomisierungsanteil a aus dem Tupel $\tilde{A}_i$ zur Basis g^r (vgl. Abschnitt 3.2) in Form von "DecryptingTupleEntry" Objekten auf dem Bulletin Board gespeichert.

Verblindung und Teilentschlüsselung in der Phase "Fake Vote Elimination": Die Tallying Authority Instanzen berechnen ein $PET(A, S)$ im verteilten Verfahren, wobei jede Instanz zuerst die Verblindung des Wertes

$$\left(\frac{A}{S}\right)^{z_i}$$

und danach für die verblindeten Werte ihr Teilresultat der Entschlüsselung berechnet. Beide Berechnungen erfolgen nach dem in den letzten beiden Paragraphen genannten Schema.

Teilentschlüsselung in der Phase "Tallying": Im letzten Schritt berechnet jede Tallying Authority ihr Teilresultat aus der Entschlüsselung der Wahl des Candidate Slates jeder abgegebenen Stimme. Auch diese Berechnung erfolgt analog der Teilentschlüsselung der "Voter Credentials".

Die Abbildung 4.4 zeigt die aus Sicht einer Tallying Authority auftretenden Methoden, wobei die dargestellte Sequenz von anderen Aktivitäten unterbrochen werden kann. Die Methoden werden also nicht in einem Durchlauf sondern in mehreren Schritten ausgeführt.

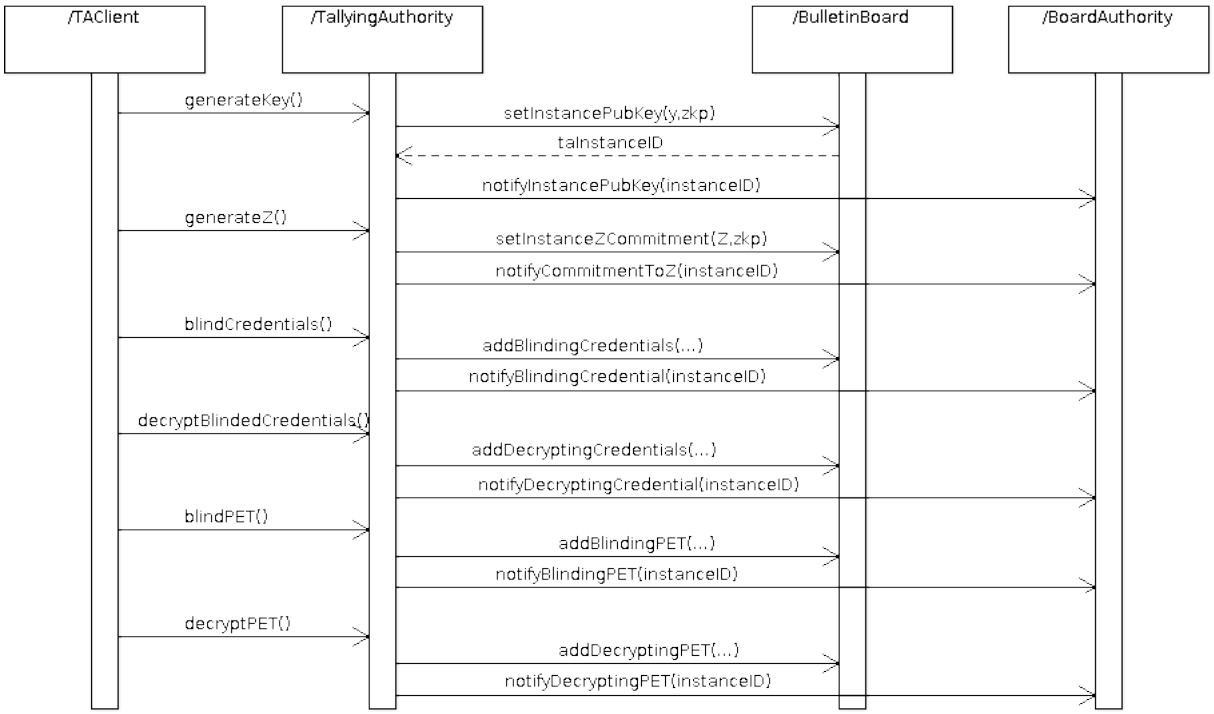


Abbildung 4.4: Auftretende Nachrichten aus Sicht einer Tallying Authority Instanz

Das Objekt "TAClient" ist im entwickelten Prototyp nicht vorhanden, es abstrahiert die angegebenen Start-Parameter des Applikationsteils "Tallying Authority" (Steuerung der auszuführenden Aktion der Tallying Authority, vgl. Abschnitt 5.1.5).

4.8 Mixing Authority

Der Begriff des Reencryption Mix-Networks wurde im Abschnitt 3.3 eingeführt. Folgend die Anforderungen, welche das zu implementierende Protokoll an ein Reencryption Mix-Network stellt:

- den Input (jeweils 3 ElGamal Tupel pro Stimme) wiederverschlüsseln
- die Reihenfolge der wiederverschlüsselten Stimmen zufällig neu ordnen
- als Output neben den wiederverschlüsselten, neu sortierten Stimmen sind zusätzlich Beweise für die korrekte Funktionsweise des Reencryption Mix-Nets vorzulegen

Während die ersten beiden Punkte verhältinismässig einfach implementiert werden können, stellt der dritte Punkt, namentlich die Erbringung der Beweise, aus mathematischer und programmiertechnischer Sicht eine grössere Herausforderung dar. Aus diesem Grund rieten die beiden Betreuer zu Beginn der Arbeit, den Einsatz der Mix-Network Library Verificatum zu prüfen. Nach umfangreichen Abklärungen wurde schliesslich entschieden, anstelle von Verificatum eine eigene Mix-Network Implementation zu verwenden. Ein ausführlicher Bericht über diesen Entscheidungsprozess findet sich im Anhang D.

Die für diese Applikation entwickelte Mix-Network Implementation unterstützt mehrere Mixing Instanzen, wobei die Anzahl der Mixing Authorities, resp. der Mixing Durchläufe konfiguriert werden kann. Die minimale Anzahl der Mixing Authorities beträgt grundsätzlich 2, da bei nur einem Mixing Durchgang die entsprechende Mixing Authority sowohl die Ausgangssituation als auch das Endresultat des gesamten Mix-Networks kennen würde und theoretisch das Abstimmungsresultat beeinflussen könnte. Da kein Threshold Verfahren eingesetzt wird, muss allerdings ohnehin vorausgesetzt werden, dass alle Parteien vertrauenswürdig sind. Aus

diesem Grund reicht in dieser Implementation bereits eine einzelne Mixing Authority aus, um die Abstimmung durchzuführen.

Die Java-Klasse "Mixer" (Implementation einer Mixing Authority) stellt folgende Methoden zur Verfügung:

- **run()**
  Die Methode fordert eine aktuelle Liste der gültigen Ballots beim Board an und schreibt diese in eine ArrayList vom Typ "BallotEntry". Anschliessend ruft sie zuerst die Reencryption Methode und danach die Shuffling Methode auf. Nach der Durchführung dieser beiden Vorgänge wird der Output (erneut eine ArrayList vom Typ "BallotEntry") zurück auf das Board geschrieben.

- **reencrypt()**
  Jedes der drei ElGamal Tupel eines Ballots wird wiederverschlüsselt. Dazu wird die im Abschnitt 4.12.1 beschriebene Klasse "ElGamalSystem" verwendet, wobei die bestehenden NIZKPs weg gelassen werden, da deren Korrektheit bereits in der Phase "Invalid Vote Elimination" überprüft wurde. Die verwendete Randomisierung wird jeweils zufällig erzeugt über den im Abschnitt 4.12.3 beschriebenen Secure Random Generator.

- **shuffle()**
  Die Position der vorgängig wiederverschlüsselten Ballots innerhalb der Liste wird zufällig vertauscht. Man spricht in diesem Zusammenhang von einer sogenannten Permutation. Nach Studium diverser Varianten wurde entschieden, die bestehende Methode "shuffle()" aus dem Package java.util.Collections zu benutzen. Da das Shuffling ein zentraler Aspekt eines Mix-Networks darstellt, findet sich die Methode im Anhang C.

Über die im Abschnitt 4.11 beschriebene Schnittstelle "Mixing Authority" kann eine Mixing Authority die Methoden "getBallots()" und "postBallots()" vom Bulletin Board aufrufen. Innerhalb dieser Methoden wird sichergestellt, dass sich das Bulletin Board im korrekten Status befindet.

4.9 Board Authority

Die Board Authority nimmt die Aufgaben einer neutralen Instanz (im Kontext eines "Distributed Key Generation" Protokolls) war. Das bedeutet, die Board Authority berechnet aus den Teilresultaten der Tallying Authorities jeweils das Gesamtresultat und legt dieses auf dem Bulletin Bord ab. Die Board Authority ist nicht in der Lage die Berechnung selbständig durchzuführen und kennt die notwendigen privaten Schlüssel nicht. Dies ändert sich auch nach der Berechnung des Gesamtresultates nicht, das heisst, auch durch die Berechnung des Gesamtresultates kann die Board Authority keine Informationen über die Teilschlüssel erlangen.

Neben den Berechnungen einer neutralen Instanz werden die Phasen "Invalid Vote Elimination" und "Ballot Replication" durch die Board Authority initiiert und durchgeführt. Beide Phasen erfordern keine Berechnung im Distributed Verfahren, also kein verteilter PET, verteiltes Entschlüsseln oder verteiltes Verblinden. Folgend eine Beschreibung, in welchen Stati des Bulletin Boards die Board Authority agiert und was ihre Aufgaben sind:

Berechnung des gemeinsamen öffentlichen Schlüssels der Tallying Authorities: Im Status "Setup" des Bulletin Boards berechnet jede Tallying Authority ihren Teilschlüssel und notifiziert die Board Authority, nachdem Sie ihren Teilschlüssel auf dem Bulletin Board abgelegt hat. Nachdem die letzte Tallying Authority dies getan hat, berechnet die Board Authority aus den Teilresultaten aller Tallying Authorities den gesamten öffentlichen Schlüssel Y mit

$$\prod_{i=1}^{n} y_i = g^{x_1} \cdot \ldots \cdot g^{x_n} = g^{x_1 + \ldots + x_n}$$

, welcher anschliessend auf dem Bulletin Board gesetzt wird. Die Berechnung stützt sich darauf, dass es n Tallying Authority Instanzen gibt und die einzelnen Teilschlüssel y_i durch jede Instanz auf dem Bulletin Board abgelegt wurde (deshalb wird die Board Authority auch von jeder Instanz notifiziert).

Berechnung der gesamten Verblindung: In den Phasen "Duplicate Vote Eliminiation" und "Fake Vote Eliminiation" wird aus den Teilverblindungen der Tallying Authorities die gesamte Verblindung berechnet. Dies geschieht nach dem selben Algorithmus wie bei der Berechnung des öffentlichen Schlüssels, da jede Tallying Authority i die Teilverblindung durch potenzieren mit ihrem Verblindungsfaktor z_i berechnet. Durch die Multiplikation aller ElGamal Tupel (Teilresultate) findet faktisch die Verblindung mit dem gemeinsamen Exponenten $z_1 + \ldots + z_n$ statt. Die Verblindung wird abhängig von der Phase auf dem "Voter Credential" (Phase "Duplicate Vote Elimination") bzw. beim PET der "Voter Credentials" (Phase "Fake Vote Elimination") durchgeführt.

Berechnung der gesamten Entschlüsselung: Nachdem die Tallying Authorities in den Phasen "Duplicate Vote Elimination", "Fake Vote Elimination" und "Tallying" ihre Teilresultate abgegeben haben, werden diese Resultate durch die Board Authority zusammengerechnet. Die Gesamtentschlüsselung wird wie folgt berechnet:

$$m = \frac{b}{\prod_{i=1}^{n} a_i}$$

Dabei ist b der Nachrichtenteil aus einem ElGamal Tupel und a_i der Randomisierungsteil potenziert mit dem privaten Schlüssel der Tallying Authority i. Durch die Multiplikation der Randomisierungsanteile a_i entsteht also der Wert

$$(g^r)^{x_1} \cdot \ldots \cdot (g^r)^{x_n} = (g^r)^{x_1 + \ldots + x_n} = g^{rx}$$

, welcher gemäss Abschnitt 3.1 zur Entschlüsselung der Nachricht führt.

Da die Board Authority neben der Kommunikation mit dem Bulletin Board über die entsprechende RMI-Schnittstelle auch Dienste für die Tallying Authorites zur Verfügung stellt, bietet die Board Authority selbst eine RMI-Schnittstelle an. Diese wird im Abschnitt 4.11 beschrieben.

4.10 Verifier

Die Aufgabe des Verifiers besteht darin, sämtliche erbrachten Resultate und Teilresultate zu überprüfen, indem er sie entsprechend nachrechnet. Die Verifier Klasse besitzt zu diesem Zweck Methoden für folgende Schritte:

- überprüfen des gemeinsam errechneten, öffentlichen Schlüssels der Tallying Authorities
- überprüfen der Beweise für die Teilschlüssel der Tallying Authorities
- überprüfen der Beweise für die Teilverblindungswerte der Tallying Authorities
- überprüfen der Beweise für die Teilverblindungen von ElGamal Tupeln
- überprüfen der Phase "Invalid Vote Elimination"
- überprüfen der Phase "Duplicate Vote Elimination"
- überprüfen der Phase "Ballot Replication"
- rudimentäre Prüfung des Reencryption Mix-Networks
- überprüfen der Phase "Fake Vote Elimination"
- überprüfen der Phase "Tallying"

Das Vorgehen innerhalb der Verifier Methoden ist stets das gleiche: Zuerst wird das ursprüngliche Resultat der Operation vom Board geladen, z.B. sämtliche Einträge vom Typ "InvalidVoteEntry". Im einem zweiten Schritt wird ein Original Set der Ursprungsdaten geladen, z.B. sämtliche Stimmen nach Abschluss der Phase "Vote Casting". Der dritte Schritt beinhaltet die nötigen Berechnungen, in diesem Beispiel die Überprüfung aller Beweise von sämtlichen Stimmen. Im vierten und letzten Schritt wird schliesslich das Resultat auf dem Board mit dem Resultat des Verifiers verglichen. Liegt eine Übereinstimmung vor, hat die Partei korrekt gehandelt.

Wichtig zu erwähnen ist, dass der Verifier grundsätzlich dieselben Operationen ausführt, wie die Parteien, welche er überprüft, aber er verfügt über eine eigene Codebasis. Dadurch wird sichergestellt, dass der Verifier

nicht fälschlicherweise denselben Code ausführt wie eine kompromittierte Partei. Die Methoden des Verifiers unterscheiden sich weiter auch darin, dass Resultate nicht auf das Bulletin Board publiziert werden, sondern lediglich für den Vergleich benutzt werden. Der Verifier darf dementsprechend auch nicht abhängig vom Board Status sein. Die Überprüfung der Originaldaten muss stets möglich sein (unabhängig der aktuellen Board Phase, sofern die Resultate bereits vorliegen). Dazu verwendet der Verifier eine eigene RMI-Schnittstelle, welche im Abschnitt 4.11 beschrieben wird.

Die Überprüfung des Reencryption Mix-Networks ist aufgrund der im Abschnitt 4.8 beschriebenen Situation nur sehr rudimentär, da die einzelnen Mixing Authorities keine Beweise zu ihrer Tätigkeit liefern. Konkret prüft der Verifier lediglich, ob eine Mixing Authority gleich viele Stimmen als Output zurückliefert, wie sie als Input erhalten hat.

Der Verifier besitzt keine eigene grafische Benutzerumgebung sondern erzeugt lediglich Ausgaben in der Konsole in Form der Logger Komponente, wobei die Protokollierungsstufe wiederum entsprechend konfiguriert werden kann (siehe Abschnitt 5.1).

4.11 RMI-Schnittstellen

Die Kommunikation unter den verschiedenen Parteien wurde mit dem Java RMI Framework realisiert. Folgende Applikationsteile agieren als RMI-Server und bieten entsprechende RMI-Schnittstellen an:

- Bulletin Board
- Registrar Authority
- Board Authority

Die Applikationsteile Registrar Authority und Board Authority agieren, wie bereits erwähnt, sowohl als RMI-Server (für ihre angebotenen Dienste) wie auch als RMI-Client für Dienste vom Bulletin Board. Eine Absicherung der Kommunikation mit Hilfe zertifikatsbasierender Authentifizierung ist nicht Gegenstand dieser Arbeit, kann aber zu einem späteren Zeitpunkt implementiert werden. Neben zusätzlicher Sicherheit würde dadurch auch eine gegenseitige Authentifikation ermöglicht, womit einzelne Schnittstellen bezüglich des Benutzerkreises eingeschränkt werden könnten.

4.11.1 Bulletin Board

Die RMI-Dienste des Bulletin Board wurden entsprechend den Rollen im Protokoll aufgetrennt. Abhängig von der Rolle des Benutzers (Wähler, Administrator, Registrar Authority, Tallying Authority, Board Authority, Mixing Authority oder Verifier) werden unterschiedliche Operationen auf dem Board zur Verfügung gestellt.

Die Abbildung 4.5 zeigt eine Übersicht der verschiedenen RMI-Schnittstellen des Bulletin Boards als Klassendiagramm. Einige globale Operationen stehen allen Rollen zur Verfügung. Diese sind im Java-Interface "IBoardServiceGeneral" enthalten. Jede rollenspezifische RMI-Schnittstelle des Bulletin Boards erweitert diese Schnittstelle und erbt somit die globalen Operationen. So kann jede Partei den aktuellen Board Status, den Abstimmungstext, das Wahlregister, die Wahloptionen, die ElGamal-Parameter, die Parameter zur Definition des Anonymitätssets sowie den öffentlichen Schlüssel der Tallying Authorities und die Anzahl TallyingAuthorities / MixingRuns abfragen. Nachfolgend werden die rollenspezifischen Schnittstellen des Bulletin Boards einzeln beschrieben.

Voter: Dem Wähler wird eine Operation für die Stimmabgabe zur Verfügung gestellt. Diese nimmt einen Eintrag vom Typ "BallotEntry" als Parameter entgegen und speichert diesen auf dem Bulletin Board ab. Damit die Methode erfolgreich ausgeführt werden kann, muss sich das Bulletin Board im Status "Vote Casting" befinden.

Admin: Dem Wahladministrator werden Operationen zur Verfügung gestellt, um die Setup-Parameter (Wahloptionen und Abstimmungstext) zu setzen, was allerdings nur im Boardstatus "Setup" und im Fall der Wahloptionen nur ein Mal möglich ist. Zwei weitere Methoden ermöglichen das Wechseln des Boardstatus und das Abfragen des Abstimmungsresultats. Vorbereitet sind zudem zwei Operationen für das Setzen der Anzahl Mixing-Durchläufe und für die Anzahl Tallying Authorities. Diese wurden allerdings nicht in das Admin GUI integriert.

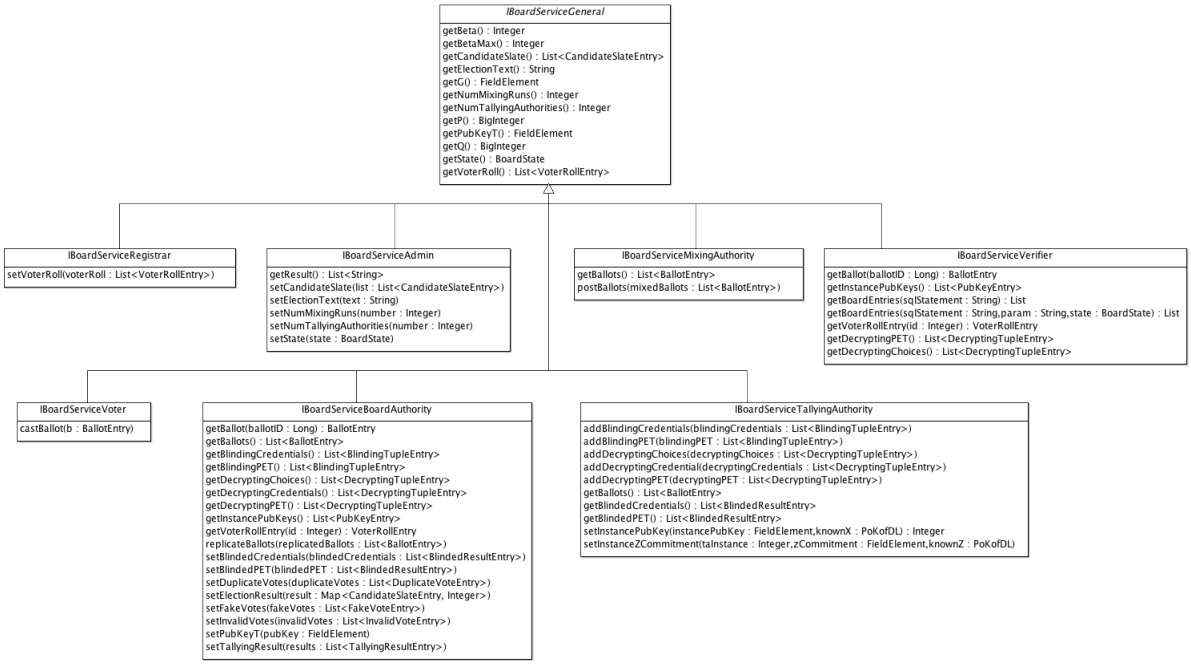


Abbildung 4.5: IBoardService Klassen

Registrar Authority: Der Registrar Authority bietet das Bulletin Board die Methode zum Setzen der Voter Roll an. Diese kann ausschliesslich im Status "Setup" gesetzt werden, muss jedoch nach der verteilten Berechnung des öffentlichen Schlüssels erfolgen (da sonst die Wählerliste nicht verschlüsselt werden kann).

Tallying Authority: Den Tallying Authorities wird vom Bulletin Board eine Schnittstelle zur Verfügung gestellt, welche Methoden für die Teilverblindung und Teilentschlüsselung anbietet. Die Tallying Authorities legen ihren öffentlichen Teilschlüssel sowie das Commitment zu ihrem Teilverblindungsfaktor auf dem Board ab und erhalten dabei die ihnen zugeteilte Instance ID. Mit Angabe dieser Instance ID meldet jede Tallying Authority via RMI-Service der Board Authority den Abschluss des jeweiligen Schrittes.

Mixing Authority: Den verschiedenen Mixing Authorities stehen im Status "Mixing Ready" eine Methode für das Lesen von Stimmen und im Status "Mixing In Progress" eine Methode für das Publizieren von durchmischten, wiederverschlüsselten Stimmen zur Verfügung.

Board Authority: Die Rolle Board Authority besitzt insgesamt am meisten Funktionalität und verfügt daher auch über 18 verschiedene Methoden, welche vom Bulletin Board angeboten werden. Die Board Authority agiert dabei, wie bereits erwähnt, in zwei Funktionen: Einerseits als neutrale Instanz, welche die Teilresultate der Tallying Authorities zum Gesamtresultat zusammenrechnet und auf dem Bulletin Board ablegt, andererseits übernimmt sie die Markierung von ungültigen Stimmen (in der Phase "Invalid Vote Elimination") und die Replikation der Stimmen (Phase "Ballot Replication"). Beide Prozesse erfordern kein verteiltes Berechnen und könnten theoretisch auch durch eine andere Partei ausgeführt werden.

Verifier: Der Verifier kann Einträge auf dem Bulletin Board lesen, damit er die erfolgten Berechnungen verifizieren kann. Neben zwei generischen Methoden für die Übergabe von JPA-Abfragen besitzt die Schnittstelle weitere Methoden, um spezifische Resultate der einzelnen Phasen abzufragen.

4.11.2 Registrar Authority

Wie das Bulletin Board bietet auch die Registrar Authority RMI-Schnittstellen für ihre Dienste an. Die zentralen Operationen sind das Publizieren der Voter Roll und die Registrierung eines Wählers. Beim Aufruf der Methode zur Registrierung eines Wählers liefert die Methode das unverschlüsselte Voter Credential als

Rückgabewert zurück.

Damit neben einer interaktiven Wahl (alle registrierten Wähler geben ihre Stimme einzeln ab) auch eine Simulation möglich ist, wurde eine zweite Schnittstelle entwickelt, welche jedoch standardmässig nicht angeboten wird. Sie kann über eine Konfigurationseinstellung aktiviert werden (siehe Abschnitt 5.5). Diese zusätzliche Schnittstelle bietet dieselben Methoden an wie die bereits beschriebene Schnittstelle, sowie zwei zusätzliche Methoden zur automatisierten / simulierten Abgabe der Stimmen für die registrierten Wähler.

4.11.3 Board Authority

Wie die Registrar Authority bietet auch die Board Authority RMI-Schnittstellen an. Grundsätzlich unterteilen sich die Operationen in zwei Gruppen: Methoden für die Notifikation der Tallying Authorities, bei welchen jeweils die Identifikation der Tallying Authority (InstanceID) mitgegeben wird und Methoden für die Durchführung der Operationen in den Phasen "Invalid Vote Elimination" und "Ballot Replication".

4.12 Kryptographie Klassen

Nachfolgend werden sogenannte Hilfsklassen erläutert, welche von den bereits beschriebenen Applikationsteilen benutzt werden. Die Gliederung der Hilfsklassen erfolgt anhand des entsprechenden Java Packages.

4.12.1 Package ch.bfh.evoting.crypto

Um kryptographische Grundfunktionen zur Verfügung zu stellen, wurden folgende Java Klassen entwickelt:

Field: Ein "Field" entspricht der Ordnung einer Gruppe. Die Gruppe wird dabei nicht weiter spezifiziert (ob multiplikativ, additiv, zyklisch, ...). Die Klasse "Field" besitzt ein einzelnes Attribut, die Ordnung. Da die Anwendung mit grossen Zahlen (üblicherweise im Bereich von 1024 bis 2048 Bits) arbeitet, wurde der Datentyp BigInteger gewählt.

FieldElement: Der Typ "FieldElement" entspricht einem Vertreter aus einer Gruppe (wobei die Ordnung über ein Element der Klasse "Field" repräsentiert wird) und besteht aus einem BigInteger, welcher den Wert definiert. Die Klasse "FieldElement" stellt Methoden für die vier Grundoperationen innerhalb von mathematischen Gruppen, sowie für das Potenzieren zur Verfügung. Zusätzlich können das additiv inverse sowie das multiplikativ inverse Element der Gruppe abgefragt werden (sofern es existiert). Ein Element dieser Klasse kann Element einer multiplikativen, zyklischen Gruppe oder einer additiven Gruppe sein oder beides.

ElGamalTuple: Ein ElGamal Tupel $c = (a, b)$ besteht jeweils aus den beiden Teilen a und b. Teil a wird "Randomization Part" und Teil b "Message Part" genannt. Die Klasse "ElGamalTuple" besitzt einen Konstruktor, der zwei Elemente (a und b) vom Typ "FieldElement" erwartet. Neben Methoden für die Ausgabe von Randomization und Message Part stehen Methoden für die Multiplikation und das Potenzieren zur Verfügung, um direkt mit dem Typ ElGamalTuple zu rechnen zu können.

ElGamalSystem: Die Klasse "ElGamalSystem" bietet die grundlegenden Funktionen für die Ver- und Entschlüsselung gemäss dem ElGamal Verfahren an, wobei die Entschlüsselung sowohl mit privatem Schlüssel als auch mit Randomisierung implementiert wurde. Eine neue Instanz von "ElGamalSystem" muss jeweils mit der Gruppenordnung p, der entsprechenden Ordnung der Untergruppe q sowie dem Generator g in der Gruppe $\mathcal{G}_q$ instanziert werden. Bei der Instanzierung wird mittels p und q der Faktor k berechnet, so dass gilt

$$p = kq + 1.$$

Bei der Konstruktion eines Objekts vom Typ "ElGamalSystem" wird geprüft, ob $k \geq 2$ gilt und, ob es sich bei p und q um gültige Primzahlen handelt. Dazu wird die Operation "isProbablePrime"[1] vom Datentyp "BigInteger" mit einer Sicherheit von 95% verwendet. Die Wahrscheinlichkeit, dass es sich wirklich um eine Primzahl handelt wird mit

$$1 - \frac{1}{2^{95}}$$

[1] Teil der Java SE: `http://docs.oracle.com/javase/7/docs/api/java/math/BigInteger.html`

berechnet. Zuletzt wird geprüft, ob ein gültiger Generator angegeben wurde. Um zu prüfen, ob ein Element g ein Generator der Gruppe q ist werden die nachfolgend aufgelisteten Eigenschaften getestet.

- Es muss gelten $g \neq 0$ und $g \neq 1$.
- Es muss gelten $g^q = 1$.
- Es darf nicht $g^k = 1$ gelten.

Schlägt eine der beschriebenen Überprüfungen fehl, wird eine entsprechende Exception geworfen, weil dadurch keine gültige Ver- oder Entschlüsselung durchgeführt werden kann. Für die Ermittlung der ElGamal-Domänen-Parameter wurde zusätzlich die Java-Klasse "GenerateCryptoParm" entwickelt, welche sich im Package "ch.bfh.evoting.tools" befindet. Diese Klasse wird nicht in der Applikation selbst verwendet, daher wird an dieser Stelle auf eine genauere Beschreibung verzichtet und stattdessen auf die Kommentare in der Javadoc und im Programmcode verwiesen.

4.12.2 Package ch.bfh.evoting.zkp

Die folgenden Zero-Knowledge Proof Klassen wurden gemäss Abschnitt 3.2 implementiert.

- Proof of Knowledge of Discrete Logarithm
- Proof of Knowledge of Equality of Discrete Logarithm
- Proof of Knowledge of Plaintext or Randomness
- Proof of Knowledge of Validity

Mit Ausnahme des Proof of Knowledge of Validity können alle Beweise sowohl interaktiv als auch nicht-interaktiv geführt werden. In letzterem Fall wird die Challenge mit dem SHA-1 Hash-Algorithmus berechnet. Um einen Proof of Knowledge of Validity zu implementieren, muss ein Beweis simuliert werden können. Diese Eigenschaft wurde bei allen implementierten Beweisen integriert.

Ausser der bereits angesprochenen Klasse Proof of Knowledge of Validity unterstützt jede Klasse die folgenden Operationen:

- generieren des Beweises:
  1. Commitment t generieren
  2. Challenge c berechnen (SHA-1-Hash)
  3. Response s berechnen
- simulieren eines Beweises:
  1. c und s zufällig bestimmen
  2. t aus c und s berechnen
- überprüfen eines Beweises:
  1. überprüfen, ob einer der Werte des Beweises fehlt
  2. vergleichen der Challenge
  3. vergleichen, ob die berechneten Werte überein stimmen
- setzen der Challenge c und des Wertes ω:
  Diese Methoden werden für interaktive Beweise bei der Stimmabgabe benutzt, da die Beweise durch einen gemeinsamen Challenge "verknüpft" werden müssen (vgl. Abschnitt 4.4).

Die konkrete Implementation ist abhängig vom entsprechenden Beweis. So besitzt ein Proof of Knowledge of Equality of Discrete Logarithms beispielsweise zwei Commitments (t_1 und t_2). Die Beweise wurden möglichst objekt-orientiert implementiert. Ein Proof of Knowledge of Plaintext or Randomness instanziert ein Proof of Knowledge of Discrete Logarithm. Ein Proof of Knowledge of Validity besteht wiederum aus mehreren Proof of Knowledge of Equality of Discrete Logarithm.

Da die Tallying Authorities jeweils beim Generieren von Teilverblindungswerten in den Phasen "Duplicate Vote Eliminiation" und "Fake Vote Elimination" einen Beweis mit drei diskreten Logarithmen erbringen müssen, wurde zusätzlich eine erweiterte Version des Proof of Knowledge of Equality of Discrete Logarithm für drei diskrete Logarithmen implementiert. Für eine Beschreibung dieser Implementation wird an dieser Stelle auf den kommentierten Programmcode verwiesen.

4.12.3 Package ch.bfh.evoting.random

"Echter" mathematischer Zufall existiert nach heutigem Wissensstand nur physikalisch in Form von radioaktivem Zerfall. Daher wird für die Ermittlung von Zufallszahlen in der Praxis oft ein grosser Aufwand betrieben und zusätzliche Hardware eingesetzt. In Absprache mit dem Auftraggeber wurde festgelegt, dass für die Generierung von Zufallswerten die Klasse "SecureRandom" aus dem Package "java.security" verwendet wird. Diese Klasse bietet einen sogenannten "Cryptographically Strong Random Number Generator" (RNG) nach dem Standard FIPS 140.2 und liefert nicht-deterministische Werte. Als Algorithmus wird "SHA1PRNG" eingesetzt, wobei PRNG für "Pseudo-Random Number Generator" steht.

Um einen späteren Austausch des Zufallsgenerators zu vereinfachen, wurden die zwei Klassen "DefaultInsecureRandom" und "DefaultSecureRandom" im Package "Random" abstrahiert. Im Falle von "DefaultInsecureRandom" wird der Standard Zufallsgenerator java.util.Random benutzt. Die Klasse "DefaultSecureRandom" generiert hingegen eine SecureRandom Instanz, deren Funktion "getRand()" Zufallszahlen zurückliefert, wobei die gewünschte Bitlänge sowie optional auch der Maximalwert festgelegt werden können.

4.13 Zusatzkomponenten

Zur Umsetzung konventioneller Aufgaben wurden bereits existierende Komponenten bzw. Frameworks eingesetzt. Zu diesen Aufgaben zählen etwa das Logging, die Verarbeitung von Start-Parametern aber auch das Testen der entwickelten Klassen. Die dazu verwendeten Java-Klassen werden nachfolgend beschrieben.

4.13.1 Logging

Die verschiedenen Applikationsteile nutzen das Apache Log4j Framework (vgl. Abschnitt 4.1), um entsprechende Statusmeldungen zu auszugeben. Jeder Applikationsteil unterscheidet zwischen verschiedenen Stufen des Loggings (im Kontext von Log4j "Log Level" genannt).

Der gewünschte Log Level des Applikationsbereiches wird beim Starten durch einen Parameter gesetzt. Die integrierten, bzw. verwendeten Log Level der entstandenen Applikation sind:

1. FATAL
   Die korrekte, weitere Ausführung des Programms ist gefährdet, bzw. unmöglich (die Applikation wird dadurch möglicherweise beendet).

2. ERROR
   Schwerwiegende Fehler, welche jedoch die weitere Ausführung des Programms erlauben.

3. WARN
   Ungewollte Zugriffe bzw. Aktionen. Die Ausführung des Programms ist weiterhin möglich.

4. INFO
   Nachrichten zum aktuellen Status der Applikation wie z.B. abgeschlossene / anstehende Aktionen.

5. DEBUG
   Detaillierte Informationen zu den Berechnungen / Phasen.

6. TRACE
   Höchster Detaillierungsgrad, wird nur bei gewissen Operationen unterstützt. Hierbei entstehen bei der Ausführung von grossen Abstimmungen (viele Wähler, hohe Beteiligung) Log-Dateien mit einer Grösse von mehreren Megabytes.

Die Log Level sind hierarchisch aufgebaut, was bedeutet, dass ein Log Level sämtliche Nachrichten seines Levels und der darüberliegenden Levels ausgibt. Standardmässig werden alle Meldungen im Terminal (Standard-Out) angezeigt, anhand eines Parameters kann jedoch auch eine Logdatei spezifiziert werden (Details siehe Abschnitt 5.1).

4.13.2 JUnit Testing

Um einzelne Teile der Applikation zu testen, wurde das Testing Framework JUnit (Version 4) eingesetzt. Der Fokus lag dabei auf den Basis-Klassen ElGamalSystem, ElGamalTuple, FieldElement und Field sowie bei den Zero-Knowledge Proofs.

Es wurden folgende JUnit Test-Klassen entwickelt:

- TestElGamalSystem
- TestElGamalTuple
- TestField
- TestFieldElement
- TestPoKofDL
- TestPoKofEqualOf3DL
- TestPoKofEqualOfDL
- TestPoKofPorR
- TestPoKofValidity

4.13.3 Datenmodell

Das Datenmodell wird grundsätzlich dynamisch durch das JPA-Modell anhand der in Abbildung 4.3 beschriebenen Java-Klassen generiert. Um die instanzierten Java-Klassen, resp. ihre Attribute in der Datenbank zu persistieren, werden die entsprechenden Klassen serialisiert.

Über den Parameter "InheritanceType" in der JPA Konfiguration wird die Strategie des Persistierungsmodells definiert. Folgende Varianten werden vom JPA Framework angeboten:

- "SINGLE_TABLE" (alle Java-Klassen werden in einer Tabelle abgebildet mit entsprechend grosser Anzahl Spalten)
- "TABLE_PER_CLASS" (alle nicht abstrakten Klassen werden in eine separate Tabelle gespeichert)
- "JOINED" (für jede Klasse wird eine Tabelle erstellt, auch wenn diese abstrakt ist)

Zu Beginn der Arbeit wurde mit der Mapping-Strategie "SINGLE_TABLE" entwickelt, gegen Ende des Projekts wurde dann auf die Strategie "TABLE_PER_CLASS" umgestellt. Das Erfüllen des Kriteriums "append-only" ist für beide Strategien umsetzbar.

Da es sich beim Registrar gemäss Protokoll Spezifikation um eine unabhängige Partei handeln muss, wird das Wahlregister (Voter Roll) in einer separaten Persistence Entity gespeichert. Die komplette Struktur der beiden Entitäten (Datenbanken) findet sich in Form eines SQL DDL-Exports der MySQL Datenbank im Anhang E.

4.13.4 Exception Handling

Die Applikation ist grundsätzlich so ausgelegt, dass allfällige Exceptions jeweils dem Aufrufer zurückgegeben werden ("throws ..." in Methodendeklaration) und erst in den GUI-Klassen bzw. Service-Starter-Klassen abgefangen werden, wo eine Rückmeldung an den Anwender erfolgt. Parallel dazu werden Fehlermeldungen von der Logger Komponente (beschrieben im Abschnitt 4.13.1) aufgezeichnet.

Die Anwendung selbst wirft folgende Typen von Exceptions:

- IllegalArgumentException
- IllegalStateException
- IllegalThreadStateException
- InvalidAlgorithmParameterException
- InvalidProofException
- UnsupportedOperationException

Besonders zu erwähnen sind dabei die Exceptions "InvalidStateException" und die durch die Applikation eingeführte Exception "InvalidProofException": Erstere wird durch vom Bulletin Board benutzt, um anzuzeigen, dass eine Operation in diesem Status nicht gültig ist. Letztere signalisiert, dass ein ungültiger Zero-Knowledge Proof verwendet wurde.

4.13.5 Dokumentation des Sourcecodes: Javadoc

Der Programmcode wurde mit Kommentaren in der Javadoc-Notation ergänzt. Die Kommentierung der Java-Klassen fand dabei nach den folgenden Grundsätzen statt:

- Javadoc Kommentare werden - analog den konventionellen Kommentaren im Programmcode - in Englisch verfasst.
- Javadoc Kommentare beschränken sich auf die Klassendefinition und auf die Methodensignatur von öffentlichen ("public") Methoden. Methoden vom Typ "private" oder "protected" werden innerhalb des Codes mit herkömmlichen Kommentaren ausreichend beschrieben.
- Attribute werden nur beschrieben sofern sinnvoll, resp. sofern deren Sinn und Zweck nicht bereits durch den Namen des Attributs ersichtlich ist.

Die durch Javadoc entstandene Dokumentation ist auf der Projektseite[2] der Origo-Plattform unter dem Dateinamen "javadoc.zip" verfügbar.

4.14 Deployment

Zur Erstellung der Javadoc und der im folgenden Kapitel beschriebenen JAR-Pakete wurde das Apache ANT Framework benutzt. Das Sourcecode-Verzeichnis wurde daher um eine ANT-Build Datei ergänzt, welche die nachfolgenden Aufgaben automatisiert ausführt:

- kompilieren der Java-Klassen
- erstellen der einzelnen JAR-Applikationsteile
- erstellen einer Zip-Datei mit dem kompletten Sourcecode (source.zip)
- erstellen einer Zip-Datei mit dem Javadoc Inhalt (javadoc.zip)

Mittels dies ANT-Builds ist es dadurch auch möglich, die Applikation weiter zu entwickeln und vor allem vereinfacht weitere Releases der Applikation zu veröffentlichen. Die ANT-Build Datei liegt im Wurzelverzeichnis des Projektcodes (innerhalb des SVN Repositories) und heisst "build.xml".

[2] `http://evoting-thesis-lb12.origo.ethz.ch/`

5 Benutzung der Applikation(en)

Die Applikation versteht sich als Software-Paket mit verschiedenen Applikationsteilen für die entsprechenden Rollen einer Abstimmung gemäss Protokoll Spezifikation. Nachfolgend wird die Installation sowie die Benutzung der Pakete bzw. der einzelnen Applikationen Schritt für Schritt beschrieben. Es wird dabei von einer konventionellen Nutzung ausgegangen. Das bedeutet, die einzelnen Schritte werden nacheinander von den verschiedenen Parteien ausgeführt. Die notwendigen Schritte zur Durchführung einer Simulation werden in Kapitel 6 beschrieben.

Systemvoraussetzungen: Wie im Abschnitt 1.3.1 erwähnt, handelt es sich um eine in Java entwickelte Lösung, welche ein installiertes "Java Runtime Environment" (JRE)[1] voraussetzt. Für die Applikationsteile "Bulletin Board" und "Registrar Authority" wird zusätzlich eine von EclipseLink / JPA unterstützte Datenbank vorausgesetzt. Die Abbildung 5.1 visualisiert die Applikationsteile und deren Voraussetzungen.

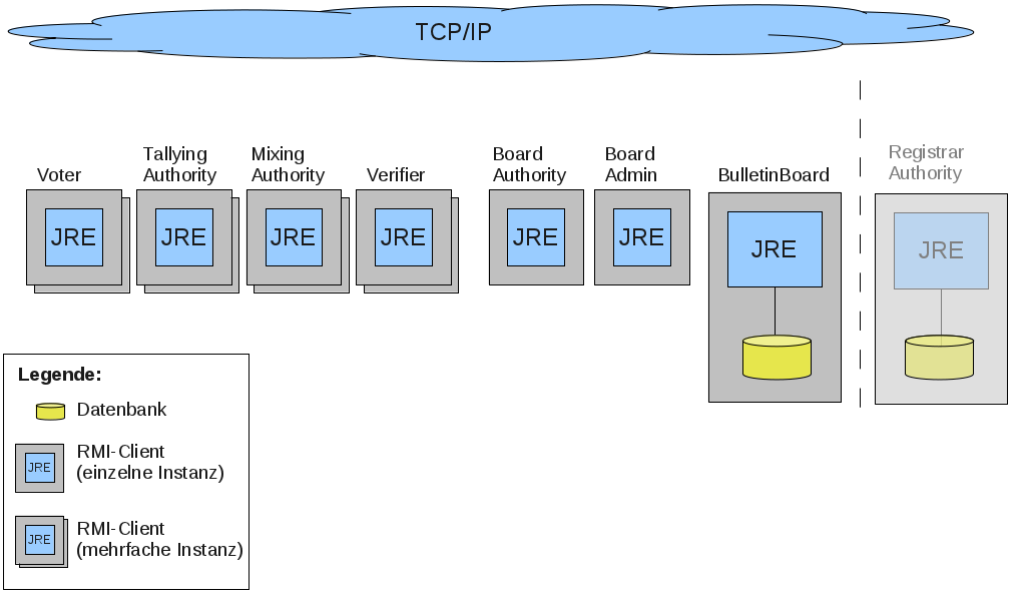


Abbildung 5.1: Systemanforderungen

Die im Abschnitt 4.11 bereits erwähnte Kommunikation über Java RMI verwendet die TCP-Ports 1099 bis 1101. Dabei handelt es sich um Standardwerte, welche über die Konfigurationsdateien geändert werden können (siehe Abschnitt 5.5). In Tabelle 5.1 werden die notwendigen TCP-Ports pro Applikationsteil aufgelistet. Diese Werte können für die Konfiguration bzw. Freischaltung von Firewall-Regeln verwendet werden.

Die Ports werden für die Einrichtung einer RMI-Registry verwendet. Um eine Ausführung der drei RMI-Serverkomponenten auf demselben Rechner zu ermöglichen wurden diesen unterschiedliche Ports zugewiesen.

5.1 Installation / Applikationsstart

Abhängig vom gewünschten Szenario sind einzelne JAR-Dateien oder die komplette Applikation (alle Teile) vom Download-Bereich des Origo-Projektes unter

`http://evoting-thesis-lb12.origo.ethz.ch/download`

[1] Download der Oracle/Sun JRE: `http://www.java.com`

Applikationsteil	eingehend	ausgehend
Voter GUI	-	1098,1101
Admin GUI	-	1098
Registrar Authority	1101	1098
Board Authority	1100	1098
Tallying Authority	-	1098,1100
Mixing Authority	-	1098
Verifier	-	1098
Bulletin Board	1098	-

Tabelle 5.1: verwendete TCP-Ports

herunterzuladen und auszuführen. Jeder Applikationsteil bietet eigene Parameter, welche beim Starten des JAR-Pakets angegeben werden können. Grundsätzlich sind keine Parameter obligatorisch, alle Applikationsteile können also ohne Parameter gestartet werden. In diesem Fall wird für unbekannte Einstellungen der Standardwert übernommen.

Neben den Parametern nutzen einige der Applikationsteile Konfigurationsdateien (Java-Properties-Dateien). Die Applikationen wurden so entwickelt, dass eine neue Datei im aktuellen Ausführungsverzeichnis der Applikation mit den notwendigen Einstellungen erstellt wird, wenn die Konfigurationsdatei nicht existiert oder nicht angegeben wurde. Mehr Informationen zu den Konfigurationsmöglichkeiten sind dem Abschnitt 5.5 zu entnehmen.

5.1.1 Bulletin Board

Installation und Start: Herunterladen und Starten der Datei "BulletinBoard.jar". Das Bulletin Board setzt eine Verbindung zu einer (MySQL) Datenbank voraus. Standardmässig nutzt die Applikation auf dem lokalen Rechner eine MySQL-Datenbank namens "test". Der Zugriff wird mit dem Benutzer "root" und leerem Passwort ausgeführt. Diese Einstellungen können über die Konfigurationsdatei des Bulletin Boards verändert werden.

Parameter:

```
-config <configfile>   use the specified configfile. if no file is
                       specified, the default file will be created and
                       used.
-help                  print this message
-logfile <logfile>     use logfile instead of logging to stdout
-loglevel <loglevel>   use the specified loglevel (apache log4j levels
                       are valid: FATAL,ERROR,WARN,INFO,DEBUG,TRACE)
-reset                 reset the BulletinBoard (clear database and save
                       states)
```

Bemerkungen: Wie bereits eingangs erwähnt ist grundsätzlich keiner der Parameter notwendig. Ohne Angabe eines Parameters startet das Bulletin Board mit den entsprechenden Standardwerten (da beim ersten Start noch keine Konfigurationsdatei existiert). Mit Hilfe des Parameters "config" kann eine Konfigurationsdatei angegeben werden (wird dieser Parameter nicht angegeben, benutzt / erstellt das Bulletin Board im aktuellen Ausführungsverzeichnis eine Datei namens "board.properties"). Der Parameter "logfile" kann benutzt werden, um die Log-Meldungen in eine Datei zu schreiben und der Parameter "loglevel" spezifiziert den zu verwendenden Log Level (Standard: "INFO", vgl. Abschnitt 4.13.1). Der Parameter "reset" wird benutzt, um die Daten der Abstimmung zu entfernen und wieder beim Status "Setup" zu beginnen (z.B. zur Durchführung einer neuen / weiteren Abstimmung).

Beispiel: Mit folgendem Befehl wird ein Bulletin Board gestartet, welches die Konfiguration aus der Datei "myboard.properties" liest, alle Log-Einträge in die Datei "myboard.log" schreibt und alle Log-Meldungen ab Log Level "DEBUG" ausgibt:

```
java -jar BulletinBoard.jar -loglevel debug -logfile myboard.log -config myboard.properties
```

5.1.2 Voter GUI

Installation und Start: Die Installation erfolgt durch Herunterladen und Ausführen der Datei "VoterGUI.jar". Dieser Applikationsteil unterstützt die Parameter:

```
-boardhost <hostname>   specify the hostname, where the bulletin board is
                        running
-boardport <port>       specify the port, where the bulletin board is
                        running
-help                   print this message
```

Bemerkungen: Ohne die Angabe der Parameter "boardhost" und "boardport" setzt die Applikation voraus, dass das Bulletin Board auf dem lokalen Rechner auf TCP-Port 1099 läuft.

Beispiel: Mit dem nachfolgenden Befehl wird ein Voter GUI gestartet, welches Verbindung zum Bulletin Board auf dem Rechner "bulletinboard.example" über den TCP-Port 1234 aufnimmt:

```
java -jar VoterGUI.jar -boardhost bulletinboard.example -boardport 1234
```

5.1.3 Admin GUI

Installation und Start: Herunterladen und Starten der Datei "AdminGUI.jar". Das Admin GUI wird analog dem Voter GUI installiert bzw. gestartet. Es benutzt dieselben Parameter ("boardhost" und "boardport"). Ebenso wie das Voter GUI verbindet das Admin GUI standardmässig auf den lokalen Rechner via Port 1099 um mit dem Bulletin Board zu kommunizieren. Die unterstützten Parameter sind:

```
-boardhost <hostname>   specify the hostname, where the bulletin board is
                        running
-boardport <port>       specify the port, where the bulletin board is
                        running
-help                   print this message
```

Beispiel: Mit dem nachfolgenden Befehl wird ein Admin GUI gestartet, welches Verbindung zum Bulletin Board auf dem Rechner "bulletinboard.example" über den TCP-Port 1234 aufnimmt:

```
java -jar AdminGUI.jar -boardhost bulletinboard.example -boardport 1234
```

5.1.4 Registrar Authority

Installation und Start: Herunterladen und Starten der Datei "Registrar.jar". Das Ausführen des Applikationsteils Registrar Authority setzt eine Datenbankverbindung voraus (analog dem Bulletin Board). Ohne die Angabe der Konfigurationsdatei (bzw. der entsprechenden Parameter) geht die Applikation davon aus, dass lokal eine MySQL-Datenbank namens "test" installiert und verfügbar ist. Der Zugriff erfolgt wiederum mit dem Benutzernamen "root" und einem leerem Passwort.

Parameter:

```
-boardhost <hostname>       specify the hostname, where the bulletinboard
                            is running (default is localhost)
-boardport <port>           specify the port, where the bulletinboard is
                            running (default is 1099)
-config <configfile>        read parameters from the specified
                            configfile.
-help                       print this message
-logfile <logfile>          use logfile instead of logging to stdout
-loglevel <loglevel>        use the specified loglevel (apache log4j
                            levels are valid:
                            FATAL,ERROR,WARN,INFO,DEBUG,TRACE)
-maxentries <number>        specify max number of entries which should be
```

```
                          included in the VoterRoll (use in combination
                          with option 'pubvoterroll', 0 means no
                          limitation)
 -printidentity <number>  print an identity file for voters. the number
                          specifies the index of voter entry (starting
                          at 1)
 -pubvoterroll            publish the VoterRoll to the BulletinBoard
 -registervoter <identity> register voter with specified identity (a
                          string) in the local registrar database
 -reset                   re-initialize / clear the registrar database
```

Bemerkungen: Neben den Parametern zum Setzen des Log Levels und der Angabe einer Log Datei ist es möglich die Voter Roll zu publizieren (Option "pubvoterroll"). Durch Spezifizieren des Parameters "maxentries" kann die Anzahl Einträge, welche in die Voter Roll geschrieben werden, begrenzt werden. Beim Publizieren der Voter Roll werden die beiden Parameter "boardhost" und "boardport" benötigt. Werden diese nicht angegeben, wird davon ausgegangen, dass das Bulletin Board auf dem Rechner localhost auf Port 1099 läuft.

Für die Stimmabgabe mit Hilfe der Applikation Voter GUI muss eine Wähleridentifikationsdatei benutzt werden. Mit Hilfe der Option "showidentity" wird der Inhalt dieser Datei für den mit Hilfe des Parameters vermerkten Wählers aus der Voter Roll erzeugt. Zudem ist es möglich, mit Hilfe des Parameters "registervoter" und der Angabe einer Wähleridentität (Zeichenfolge) einen Wähler zu registrieren.

Beispiel: Nachfolgend ein Beispiel eines Aufrufs der Registrar Authority, welche

- den Wähler mit der Identifikation "testuser01@mydomain" registriert,
- den Inhalt der Wähleridentifikationsdatei für den ersten registrierten Wähler anzeigtund
- die Voter Roll publiziert

```
java -jar Registrar.jar -reset -registervoter user01@mydomain -printidentity 1 -pubvoterroll
0 [main] INFO Registrar  - Loglevel set to 'INFO'
6 [main] INFO Registrar  - clearing database...
762 [main] INFO Registrar  - RMI Registry created on port 1101
965 [main] INFO Registrar  - RMI Service 'RegistrarIdentity' binded
1360 [main] INFO Registrar.Service  - voter 'testuser01@mydomain' registred
1360 [main] INFO Registrar.Service  - generating voter identity file...
1360 [main] INFO Registrar.Service  - content of voter identity file:
1
user01@mydomain
97658761931075...
1361 [main] INFO Registrar.Service  - copy this content above to a...
1458 [main] INFO Registrar.Service  - generating encrypted voter...
1470 [main] INFO Registrar.Service  - encryptied voter credentials built...
1534 [main] INFO Registrar.Service  - voter roll published to board.
1534 [main] INFO Registrar  - VoterRoll published to BulletinBoard
```

Die Ausgabe des Parameters "printidentity" kann in eine Datei kopiert werden, welche bei der interaktiven Stimmabgabe mit dem Voter GUI zur Identifikation benutzt werden kann (vgl. Abschnitt 5.4). Das Format der Wähleridentifikationsdatei ist:

- 1. Zeile: ID des Voter Roll Eintrags (wird benutzt im Anonymitätsset)
- 2. Zeile: Wähleridentifikation
- 3. Zeile: Wählergeheimnis (Sigma), das benutzt wird zur Abstimmung

5.1.5 Tallying Authority

Installation und Start: Herunterladen und Starten der Datei "TallyingAuthority.jar". Das Ausführen erfordert eine Verbindung zum Bulletin Board und zur Board Authority. Gültige Parameter sind:

```
-authorityhost <hostname>   specify the hostname, where the
                            boardauthority is running
-authorityport <port>       specify the port, where the boardauthority is
                            running
-blindCred                  blind credentials posted by the voters
                            (DuplicateVoteElimination)
-blindPET                   blind PET of voter credentials / voter roll
                            entry (FakeVoteElimination)
-boardhost <hostname>       specify the hostname, where the bulletinboard
                            is running
-boardport <port>           specify the port, where the bulletinboard is
                            running
-config <configfile>        use the specified configfile. if no file is
                            specified, the default file will be created
                            and used.
-decryptChoices             decrypt the voter choices for tallying and
                            post partial result (Tallying)
-decryptCred                decrypt prior blinded credentials and post
                            partial result (DuplicateVoteElimination)
-decryptPET                 decrypt prior blinded PET values and post
                            partial result (FakeVoteElimination)
-help                       print this message
-init                       initialize. means build and publish keys and
                            blining factor (-commitment).
-logfile <logfile>          use logfile instead of logging to stdout
-loglevel <loglevel>        use the specified loglevel (apache log4j
                            levels are valid:
                            FATAL,ERROR,WARN,INFO,DEBUG,TRACE)
```

Bemerkungen: Die möglichen Tätigkeiten entsprechen den Aktionen innerhalb der angegebenen Phase. Der Ablauf aus Sicht einer Tallying Authority ist gegliedert in:

1. generieren und publizieren von Teilschlüssel und Teilverblindungsfaktor inkl. dem entsprechenden Commitment (Parameter "init")
2. Teilverblinden der Voter Credentials (Parameter "blindCred")
3. Teilentschlüsseln der verblindeten Voter Credentials (Parameter "decryptCred")
4. Teilverblinden der $PET(A, S)$ Werte aller Stimmen (Parameter "blindPET")
5. Teilentschlüsseln der verblindeten $PET(A, S)$ Werte (Parameter "decryptPET")
6. Teilentschlüsseln der abgegebenen Stimmen (Parameter "decryptChoices")

Aus praktikablen Gründen wird die Anwendung nach den Berechnungen in der Phase "Setup" beendet. In der Phase "Duplicate Vote Elimination" wird die Tallying Authority mit den entsprechenden Parametern wieder gestartet, dasselbe gilt für die Phasen "Fake Vote Elimination" und "Tallying". Die Aktivitäten einer Tallying Authority erfolgen somit in sechs verschiedenen Aufrufen.

Beispiel:

```
java -jar TallyingAuthority.jar -init
...
java -jar TallyingAuthority.jar -blindCred
java -jar TallyingAuthority.jar -decryptCred
```

```
...
java -jar TallyingAuthority.jar -blindPET
java -jar TallyingAuthority.jar -decryptPET
...
java -jar TallyingAuthority.jar -decryptChoices
```

Die Tallying Authority speichert nach der Initialisierung ihren privaten Schlüssel und den Verblindungsfaktor in der Konfigurationsdatei (in diesem Beispiel die Standard-Konfigurationsdatei "tallyingauthority.properties").

5.1.6 Board Authority

Installation und Start: Herunterladen und Starten der Datei "BoardAuthority.jar". Die gültigen Parameter sind:

```
-boardhost <hostname>   specify the hostname, where the bulletinboard is
                        running
-boardport <port>       specify the port, where the bulletinboard is
                        running
-checkinvalidvotes      start the invalid vote elimination phase
-help                   print this message
-logfile <logfile>      use logfile instead of logging to stdout
-loglevel <loglevel>    use the specified loglevel (apache log4j levels
                        are valid: FATAL,ERROR,WARN,INFO,DEBUG,TRACE)
-replicateballots       start the ballot replication phase
```

Bemerkungen: Wie im Abschnitt 4.9 beschrieben, übernimmt die Board Authority sowohl die Berechnung des Gesamtresultats der Tallying Authorities als auch die Durchführung der Operationen in den Phasen "Invalid Vote Elimination" und "Ballot Replication". Dies bedingt, dass die Applikation bei ihrem ersten Aufruf (ohne Parameter) nach Abschluss der Phase "Setup" manuell beendet werden muss (mittels System Interrupt Ctrl+C). In der Phase "Invalid Vote Elimination" wird sie erneut gestartet mit dem Parameter "checkinvalidvotes" und läuft weiter bis zum Abschluss der Phase "Duplicate Vote Elimination". Bevor die Phase "Ballot Replication" gestartet wird, muss die Applikation wiederum beendet und mit dem Parameter "replicateballots" neu gestartet werden.

Beispiel:

```
java -jar BoardAuthority.jar
...
[manuelles beenden mit Ctrl+C]
java -jar BoardAuthority.jar -checkinvalidvotes
...
[manuelles beenden mit Ctrl+C]
java -jar BoardAuthority.jar -replicateballots
```

Die hier aufgelisteten Befehle führen alle Aktionen einer Board Authority während einer Abstimmung aus.

5.1.7 Mixing Authority

Installation und Start: Herunterladen und Ausführen der Datei "MixingAuthority.jar". Gültige Parameter sind:

```
-boardhost <hostname>   specify the hostname, where the bulletinboard is
                        running
-boardport <port>       specify the port, where the bulletinboard is
                        running
-config <configfile>    use the specified configfile. if no file is
                        specified, the default file will be created and
                        used.
-help                   print this message
```

```
-logfile <logfile>      use logfile instead of logging to stdout
-loglevel <loglevel>    use the specified loglevel (apache log4j levels
                        are valid: FATAL,ERROR,WARN,INFO,DEBUG,TRACE)
```

Bemerkungen: Analog den anderen Applikationsteilen, kann auch hier die Konfigurationsdatei, der Log Level und eine Log Datei angegeben werden. Die Mixing Authority ist im Unterschied zur Tallying Authority nur in einer Phase beteiligt, die Angabe einer Tätigkeit entfällt also.

Beispiel:

```
java -jar MixingAuthority.jar -boardhost bulletinboard.example -boardport 1234
```

5.1.8 Verifier

Installation und Start: Herunterladen und Ausführen der Datei "Verifier.jar". Gültige Parameter sind:

```
-boardhost <hostname>   specify the hostname, where the bulletinboard is
                        running
-boardport <port>       specify the port, where the bulletinboard is
                        running
-help                   print this message
-logfile <logfile>      use logfile instead of logging to stdout
-loglevel <loglevel>    use the specified loglevel (apache log4j levels
                        are valid: FATAL,ERROR,WARN,INFO,DEBUG,TRACE)
```

Bemerkungen: Analog den anderen Applikationsteilen, kann auch hier die Konfigurationsdatei, der Log Level und eine Log Datei angegeben werden.

Beispiel:

```
java -jar Verifier.jar -boardhost bulletinboard.example -boardport 1234
```

5.2 Durchführung einer Abstimmung

Die folgenden Schritte beschreiben den regulären Ablauf einer Abstimmung mit sämtlichen, teilnehmenden Parteien. Es werden Standardwerte angenommen, auf die Angabe von optionalen Parametern wird verzichtet.

1. **Bulletin Board:** Starten der Bulletin Board Instanz:

   ```
   java -jar BulletinBoard.jar
   ```

   Mit der Ausgabe der Log Meldungen

   ```
   1441 [main] INFO Board  - RMI Service 'BoardServiceRegistrar' binded
   1450 [main] INFO Board  - RMI Service 'BoardServiceVoter' binded
   1459 [main] INFO Board  - RMI Service 'BoardServiceAdmin' binded
   1464 [main] INFO Board  - RMI Service 'BoardServiceMixingAuthorities' binded
   1471 [main] INFO Board  - RMI Service 'BoardServiceTallyingAuthority' binded
   1480 [main] INFO Board  - RMI Service 'BoardServiceAuthority' binded
   1488 [main] INFO Board  - RMI Service 'BoardServiceVerifier' binded
   ```

   ist die Initialisierung abgeschlossen und das Bulletin Board bereit (der Applikationsteil läuft weiter). Es wird davon ausgegangen, dass das Bulletin Board zum ersten Mal gestartet wurde, falls nicht, muss mit dem Parameter "reset" das Bulletin Board reinitialisiert werden (Schritt 3, die Schlüsselgenerierung der Tallying Authorities entfällt dabei).

2. **Board Authority:** Starten der Board Authority Instanz:

```
java -jar BoardAuthority.jar
```

Mit der Ausgabe der Log Meldung

```
141 [main] INFO BoardAuthority  - RMI Service 'BoardAuthority' binded
```

ist die Initialisierung der Board Authority abgeschlossen (der Applikationsteil läuft weiter).

3. **Tallying Authorities:** Jede Tallying Authority generiert und publiziert zuerst Teilschlüssel und Teilverblindungsfaktor. Wir verwenden in diesem Beispiel zwei Tallying Authorities (Standardwert) mit den Konfigurationsdateien "tallyingauthority1.properties" und "tallyingauthority2.properties":

   ```
   java -jar TallyingAuthority.jar -init
   ```

   Nachdem die erste Tallying Authority ihre Initialisierung abgeschlossen hat, wird die erstellte Konfigurationsdatei "tallyingauthority.properties" in "tallyingauthority1.properties" umbenannt. Es folgt das Initiieren der zweiten Tallying Authority mit dem gleichen Befehl und anschliessendem Umbenennen der Konfigurationsdatei von "tallyingauthority.properties" in "tallyingauthority2.properties".

   Sobald die letzte Tallying Authority ihre Teilresultate publiziert hat, berechnet die Board Authority automatisch den gemeinsamen, öffentlichen Schlüssel. Nach diesem Schritt kann die Board Authority beendet werden.

4. **Registrar Authority:** Starten der Registrar Authority, Registrieren von zwei Wählern und Ausgeben der Wähleridentifikationsdatei für den ersten Wähler. Anschliessend Publizieren des Wahlregisters (Voter Roll) auf das Bulletin Board:

   ```
   java -jar Registrar.jar -registervoter "max.muster@example.net" -printidentity 1
   java -jar Registrar.jar -registervoter "hans.meier@example.net" -pubvoterroll
   ```

   Auch hier wird davon ausgegangen, dass die Registrar Authority zum ersten Mal gestartet wird, bzw. keine Wähler in einer vorderen Abstimmung registriert wurden. Andernfalls kann die Registrierung der Wähler (Parameter "registervoter") ausgelassen werden.

   Der erste Befehl registriert einen Wähler und gibt die dazugehörige Wähleridentifikationsdatei aus. Beim zweiten Aufruf wird ein weiterer Wähler registriert und die Wählerliste auf das Bulletin Board publiziert. Es ist zu beachten, dass die Grösse der Wählerliste mindestens der Grösse des Bulletin Board Parameters "BETA" entsprechen muss und, sofern gesetzt, auch mindestens der Grösse des Bulletin Board Parameters "BETAMAX". Die Bedeutung der beiden Parameter wird ausführlich im Abschnitt 6.2 erklärt.

5. **Administrator:** Starten des Admin GUIs und Setzen des Abstimmungsnamen sowie der Wahloptionen:

   ```
   java -jar AdminGUI.jar
   ```

   Sichern der eingetragenen Werte auf das Bulletin Board durch Klick auf "save setup parameters", danach Überführen des Board Status von "Setup" nach "Vote Casting" durch Klick auf "next phase". Details zur Benutzung des Admin GUIs finden sich im Abschnitt 5.3. Das Admin GUI könnte nach dem Statuswechsel theoretisch wieder beendet werden.

6. **Wähler:** Abgabe der Stimme über das Voter GUI:

   ```
   java -jar VoterGUI.jar
   ```

   Dazu ist zuerst die Identifikation vorzunehmen (auswählen der im Schritt 4 erstellten Wähleridentifikationsdatei), danach kann mittels Klick auf "cast ballot" die Stimme abgegeben werden (Details, siehe Abschnitt 5.4).

7. **Administrator:** Überführen des Abstimmungsstatus von "Vote Casting" nach "Invalid Vote Elimination" mittels Admin GUI.

8. **Board Authority:** Beenden der Board Authority (falls diese nicht bereits im Schritt 3 beendet wurde) und initiieren der Phase "Invalid Vote Elimination".

   ```
   ... INFO BoardAuthority.Service  - the last tallying authority has published...
   [manuelles beenden mit Ctrl+C]
   java -jar BoardAuthority.jar -checkinvalidvotes
   ```

   Nach diesem Schritt darf die Board Authority nicht beendet werden (oder sie wird in Schritt 10 wieder gestartet).

9. **Administrator:** Überführen des Abstimmungsstatus von "Invalid Vote Elimination" nach "Duplicate Vote Elimination".

10. **Tallying Authorities:** Jede Tallying Authority generiert Teilverblindungen der Voter Credentials:

    ```
    java -jar TallyingAuthority.jar -config tallyingauthority1.properties -blindCred
    java -jar TallyingAuthority.jar -config tallyingauthority2.properties -blindCred
    ```

    Sobald die letzte Tallying Authority ihre Teilresultate publiziert hat, berechnet die Board Authority automatisch das Gesamtresultat. Danach werden die Teilentschlüsselungen jeder Tallying Authority durchgeführt:

    ```
    java -jar TallyingAuthority.jar -config tallyingauthority1.properties -decryptCred
    java -jar TallyingAuthority.jar -config tallyingauthority2.properties -decryptCred
    ```

    Nachdem die letzte Tallying Authority ihr Teilresultat auf dem Bulletin Board publiziert hat, wird die Gesamtentschlüsselung berechnet und mehrfache Stimmabgaben markiert.

11. **Administrator:** Überführen des Abstimmungsstatus von "Duplicate Vote Elimination" nach "Ballot Replication".

12. **Board Authority:** Beenden der Board Authority Instanz. Anschliessend neu starten, um die Replikate für alle noch gültigen Stimmen zu generieren:

    ```
    ...INFO BoardAuthority.Service  - duplicate vote elimination finished
    [manuelles beenden mit Ctrl+C]
    java -jar BoardAuthority.jar -replicateballots
    ```

13. **Administrator:** Überführen des Abstimmungsstatus von "Ballot Replication" nach "Mixing Ready".

14. **Mixing Authorities:** Sequentielles Starten der Mixing Authority Instanzen zwecks Reencryption und Shuffling. Standardmässig erwartet das Bulletin Board drei Mixing-Durchläufe:

    ```
    java -jar MixingAuthority.jar
    java -jar MixingAuthority.jar
    java -jar MixingAuthority.jar
    ```

15. **Administrator:** Überführen des Abstimmungsstatus von "Mixing Ready" nach "Fake Vote Elimination".

16. **Tallying Authorities:** Jede Tallying Authority generiert Teilverblindungen der Operation $PET(A, S)$:

    ```
    java -jar TallyingAuthority.jar -config tallyingauthority1.properties -blindPET
    java -jar TallyingAuthority.jar -config tallyingauthority2.properties -blindPET
    ```

    Nachdem die Board Authority die Gesamtverblindung berechnet und publiziert hat, werden die Resultate verteilt entschlüsselt:

    ```
    java -jar TallyingAuthority.jar -config tallyingauthority1.properties -decryptPET
    java -jar TallyingAuthority.jar -config tallyingauthority2.properties -decryptPET
    ```

Sobald die letzte Tallying Authority ihre Teilresultate publiziert hat, berechnet die Board Authority automatisch die Gesamtentschlüsselung und markiert gefälschte Stimmen.

17. **Administrator:** Überführen des Abstimmungsstatus von "Fake Vote Elimination" nach "Tallying".

18. **Tallying Authorities:** Jede Tallying Authority führt eine Teilentschlüsselung der zu diesem Zeitpunkt noch gültigen Stimmen durch:

    ```
    java -jar TallyingAuthority.jar -config tallyingauthority1.properties -decryptChoice
    java -jar TallyingAuthority.jar -config tallyingauthority2.properties -decryptChoice
    ```

    Sobald die letzte Tallying Authority ihre Teilresultate publiziert hat, berechnet die Board Authority automatisch das Gesamtresultat der Abstimmung.

19. **Administrator:** Mittels Klick auf "refresh" kann das Abstimmungsresultat im Feld "statistic" eingesehen werden.

20. **Verifier:** Ein Verifier kann nun zwecks Überprüng der gesamten Abstimmung die Berechnungen wiederholen und mit dem Resultat auf dem Bulletin Board vergleichen:

    ```
    java -jar Verifier.jar
    ```

5.3 Admin GUI

Die Administrationskonsole dient dem Wahlleiter dazu, die Abstimmung einzurichten und jeweils in die Zustände zu überführen, welche im Abschnitt 4.3 eingeführt wurden. Abbildung 5.2 zeigt die Abstimmung nach Abschluss der "Vote Authorization" in der Phase "Tallying".

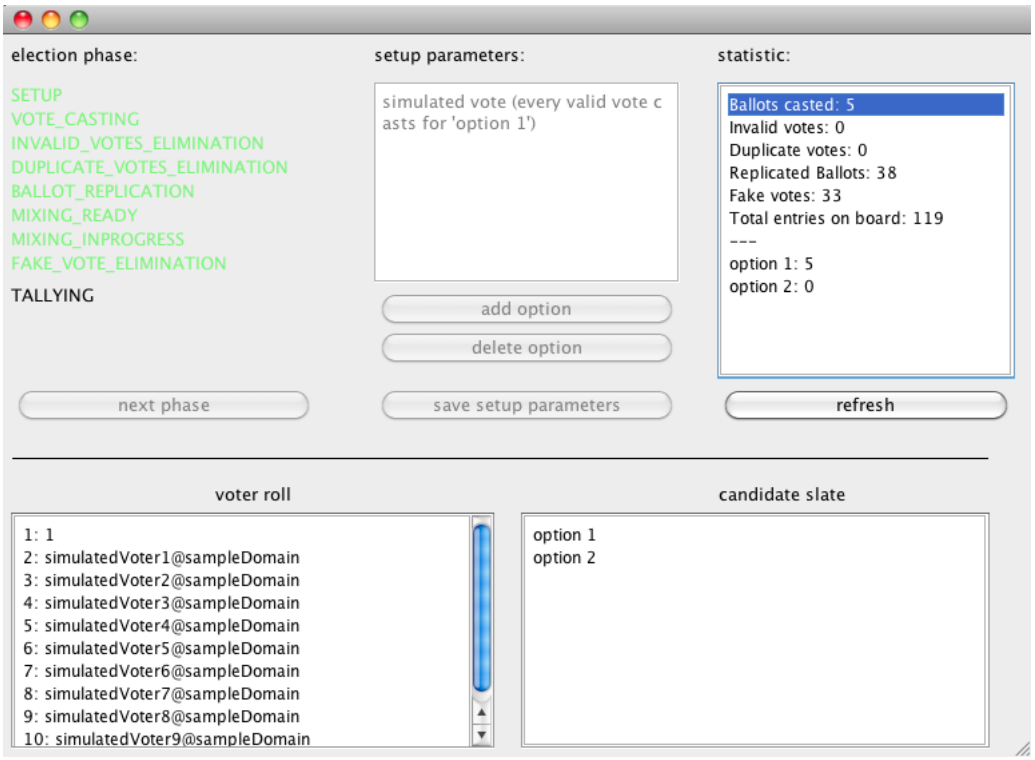


Abbildung 5.2: Admin GUI im Status "Tallying"

Abgeschlossene Phasen werden grün dargestellt, pendente Phasen grau und die aktive Abstimmungsphase schwarz. Um das Bulletin Board vom Initialzustand "Setup" in den nächsten Zustand "Vote Casting" zu überführen, müssen die folgenden Bedingungen erfüllt sein:

- die Abstimmungsfrage wurde gesetzt
- die Wahloptionen wurden festgelegt
- der öffentliche Schlüssel wurde durch die Board Authority auf dem Bulletin Board gesetzt
- die Wählerliste (Voter Roll) wurde durch die Registrar Authority publiziert

Die Stimmabgabe kann nur während der Phase "Vote Casting" erfolgen. Der Zeitraum der Stimmabgabe wird daher durch den Administrator gesteuert.

Die gewählte Umsetzungsvariante bedingt absichtlich, dass die verschiedenen Parteien miteinander kommunizieren. Setzt der Administrator beispielsweise das Bulletin Board in den Zustand "Invalid Vote Elimination", startet die erste Phase der "Vote Authorization" nicht automatisch, sondern erst durch die Partei "Board Authority", welche die entsprechende Methode ausführt. Die Statistik am rechten Rand der Administrationskonsole wird nach jeder Phase aktualisiert. Eine manuelle Aktualisierung ist über die entsprechende Schaltfläche auf der rechten Seite möglich.

5.4 Voter GUI

Die Applikation für den Wähler ist absichtlich sehr rudimentär aufgebaut. Abbildung 5.3 zeigt einen entsprechenden Screenshot.

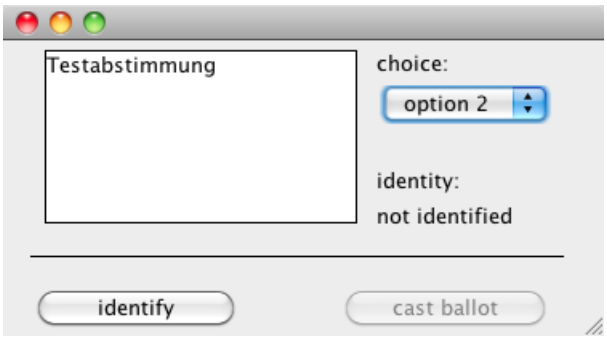


Abbildung 5.3: VoterGUI

Der Wähler sieht die aktuelle Abstimmungsfrage und in einer Dropdown-Liste daneben die möglichen Antworten. Um keine Wahloption zu bevorzugen, ist die Auswahl standardmässig leer.

Bevor der Wähler seine Stimme abgeben kann, muss er sich entsprechend authentifizieren. Wie im Use Case in Anhang A.4 beschrieben, gehen wir davon aus, dass sich der Wähler vorgängig bei der Wahlbehörde registriert hat und über einen digitalen Stimmrechtsausweis in Form einer Wähleridentifikationsdatei (Erstellung siehe Abschnitt 5.1.4) verfügt. Die Daten dieser Datei (ID und Name des Wählers sowie sein persönlicher, geheimer Abstimmungsausweis σ) werden eingelesen, der Name des Wählers wird anschliessend in der Applikation unter "Identität" angezeigt. Durch Drücken der Schaltfläche "cast ballot" wird nun der im Abschnitt 4.4 beschriebene Prozess der Stimmabgabe gestartet und der Wähler kann seine Stimme auf dem Bulletin Board platzieren. Der Wähler erhählt anschliessend ein entsprechendes Dialogfenster, welches die erfolgreiche Stimmabgabe bestätigt oder aber einen Fehler (gemäss Abschnitt 4.13.4) zurückliefert. Die Applikation wird nach Bestätigung des Dialogfensters automatisch geschlossen.

5.5 Konfiguration

Die Applikationsteile Bulletin Board, Registrar Authority, Board Authority und Tallying Authority können über Java Property Files konfiguriert werden. Jeder Applikationsteil erstellt, wie im Abschnitt 5.1 bereits erwähnt, automatisch eine Standard-Konfiguration, sofern keine Datei angegeben wurde bzw. die Standard-Konfigurationsdatei nicht existiert.

Nachfolgend die Auflistung der möglichen Parameter pro Applikationsteil. Mit einem Stern (*) markierte Parameter werden durch die Applikation selbst verwaltet und sollten nicht verändert werden.

Bulletin Board: Die Standard-Konfigurationsdatei heisst "board.properties", definierte Parameter sind:

- BETA
  Der Sicherheits-Parameter Beta legt fest, wie viele Identitäten das Anonymitätsset einer Stimme minimal beinhalten muss.
  Standardwert: 2

- BETAMAX
  Der Sicherheits-Parameter Beta kann durch Setzen eines Werts von BETAMAX > 0 begrenzt werden. Standardmässig ist diese Begrenzung nicht gesetzt. Die Anzahl der in der Phase "Ballot Replication" replizierten Stimmen pro abgegebener Stimme kann daher zwischen BETA und der Anzahl Wähler der Voter Roll variieren.
  Standardwert: 0 (nicht begrenzt)

- *BOARDSTATE
  Der aktuelle Boardstatus. Über diesen Wert speichert die Bulletin Board Instanz, in welchem Status sie sich befindet.
  Standardwert: SETUP

- *ELECTIONTEXT
  Über diesen Wert speichert das Bulletin Board den Abstimmungstext.
  Standardwert: -

- G
  Der Wert des ElGamal Domänen-Parameters g. Bei g muss es sich um einen gültigen Generator der Gruppe handeln (siehe Abschnitt 4.12.1).
  Standardwert: (Generator der Gruppe q)

- JPA_DRIVER
  Dieser Wert konfiguriert zusammen mit dem Parameter "JPA_URL" die Datenbankverbindung. Mittels Angabe des Treibers wird das benutzte Eclipselink (JPA) Framework konfiguriert.
  Standardwert: com.mysql.jdbc.Driver

- JPA_PASSWD
  Das Passwort für die Verbindung zur Datenbank.
  Standardwert: -

- JPA_URL
  Dieser Wert konfiguriert zusammen mit dem Parameter "JPA_DRIVER" die Datenbankverbindung. Mittels Angabe der URL wird der Rechner, Port und Datenbank-Name spezifiziert. Der Standardwert konfiguriert eine MySQL-Datenbank auf dem lokalen Rechner (TCP-Port 3306) und der Datenbank "Test".
  Standardwert: jdbc:mysql://localhost:3306/test

- JPA_USER
  Der Benutzername für die Verbindung zur Datenbank.
  Standardwert: root

- LOCAL_RMI_REGISTRY_PORT
  Die Portnummer, auf welcher die RMI-Registry für das Bulletin Board gestartet wird.
  Standardwert: 1099

- *NUMCOMPLETEDMIXINGRUNS
  Mit diesem Wert speichert das Bulletin Board die Anzahl bereits durchgeführter Mixing-Durchläufe.
  Standardwert: 0

- NUMIXINGRUNS
  Die Anzahl der Mixing-Durchgänge durch die Mixing Authorities, also die Anzahl Mixing Authorities.
  Standardwert: 3
- *NUMREGISTEREDTALLYINGAUTHORITIES
  Mit diesem Wert speichert das Bulletin Board die Anzahl bereits registrierter Tallying Authorites (Tallying Authorities, welche die Initialisierung abgeschlossen haben).
  Standardwert: 0
- NUMTALLYINGAUTHORITIES
  Die Anzahl der an der Wahl teilnehmenden Tallying Authorities.
  Standardwert: 2
- P
  Der Wert des ElGamal Domänen-Parameters p. Der Wert p muss eine Primzahl sein und aus $p = k*q+1$ gebildet sein (siehe Abschnitt 4.12.1).
  Standardwert: (1024-Bit Wert)
- *PUBKEYT
  Mit diesem Wert speichert das Bulletin Board den verteilt berechneten, öffentlichen Schlüssel der Tallying Authorities.
  Standardwert: -
- Q
  Der Wert des ElGamal Domänen-Parameters q. Der Wert q muss eine Primzahl sein (siehe Abschnitt 4.12.1).
  Standardwert: (160-Bit Wert)

Board Authority: Die Konfigurationsdatei der Board Authority heisst "boardauthority.properties". Sie nutzt die folgenden Parameter:

- BOARD_RMI_HOSTNAME
  Mit diesem Wert wird der Rechnername des Bulletin Boards konfiguriert. Dieser Wert kann auch als Start-Parameter übergeben werden (vgl. Abschnitt 5.1).
  Standardwert: localhost
- BOARD_RMI_PORT
  Dieser Wert spezifiziert den Port, auf welchem der RMI-Service des Bulletin Boards registriert wurde.
  Standardwert: 1099
- LOCAL_RMI_REGISTRY_PORT
  Mit diesem Wert wird festgelegt, auf welchem lokalen Port der RMI-Service der Board Authority registriert werden soll.
  Standardwert: 1100

Registrar Authority: Die Konfigurationsdatei der Registrar Authority heisst "registrar.properties". Die möglichen Parameter sind:

- JPA_DRIVER
  Dieser Wert konfiguriert zusammen mit dem Parameter "JPA_URL" die Datenbankverbindung. Mittels Angabe des Treibers wird das benutzte Eclipselink (JPA) Framework konfiguriert.
  Standardwert: com.mysql.jdbc.Driver
- JPA_PASSWD
  Das Passwort für die Verbindung zur Datenbank.
  Standardwert: -
- JPA_URL
  Dieser Wert konfiguriert zusammen mit dem Parameter "JPA_DRIVER" die Datenbankverbindung. Mittels Angabe der URL wird der Datenbank-Rechner, Port und Datenbank-Name spezifiziert. Der Standardwert konfiguriert eine MySQL-Datenbank auf dem lokalen Rechner (TCP-Port 3306) und der

Datenbank “Test”.
Standardwert: jdbc:mysql://localhost:3306/test

- JPA_USER
  Der Benutzername für die Verbindung zur Datenbank.
  Standardwert: root
- LOCAL_RMI_REGISTRY_PORT
  Mit diesem Wert wird festgelegt, auf welchem lokalen Port der RMI-Service der Registrar Authority registriert werden soll.
  Standardwert: 1101
- SIMULATION_ALLOWED
  Dieser Wert legt fest, ob zusätzlich die RMI-Schnittstelle exportiert wird, welche das Simulieren einer Wahl erlaubt (vgl. Abschnitt 4.11.2). Gültige Werte sind “true” oder “false”.
  Standardwert: false

Tallying Authority: Die Konfigurationsdatei der Tallying Authority heisst standardmässig “tallyingauthority.properties” und hat die folgenden Parameter definiert:

- BOARDAUTHORITY_RMI_HOSTNAME
  Dieser Wert definiert einen gültigen Hostnamen (DNS-Name, Hosts-Eintrag oder IP-Adresse) des Computers, auf welchem die Board Authority ausgeführt wird.
  Standardwert: localhost
- BOARDAUTHORITY_RMI_PORT
  Dieser Wert definiert einen gültigen Port, auf welchem der Computer der Board Authority die RMI-Registry instanziert hat.
  Standardwert: 1100
- BOARD_RMI_HOSTNAME
  Dieser Wert definiert einen gültigen Hostnamen (DNS-Name, Hosts-Eintrag oder IP-Adresse) des Computers, auf welchem das Bulletin Board ausgeführt wird.
  Standardwert: localhost
- BOARD_RMI_PORT
  Dieser Wert definiert einen gültigen Port, auf welchem der Computer des Bulletin Boards die RMI-Registry instanziert hat.
  Standardwert: 1099
- *INSTANCE_NO
  Mit dieser Einstellung wird die vom Bulletin Board vergebene Instanz-Nummer gespeichert. Jede Tallying Authority kann während des gesamtem Abstimmungsprozesses mehrmals gestartet / beendet werden.
  Standardwert: -
- *PRIVATE_KEY
  Mit dieser Einstellung wird der von der Tallying Authority Instanz benutzte, private Teilschlüssel gespeichert.
  Standardwert: -
- *Z
  Mit dieser Einstellung wird der von der Tallying Authority Instanz benutzte Teilverblindungsfaktor z gespeichert.
  Standardwert: -

6 Simulation einer Wahl

Dieses Kapitel umfasst die Installation und Benutzung der Simulationskomponente, sowie die exemplarische Beschreibung eines Simulationsdurchlaufs.

Neben den einzelnen Applikationsteilen, welche zur Durchführung einer Wahl benötigt werden, ist auch eine Komponente zur Durchführung einer Simulation entwickelt worden. Diese vereint die notwendigen Funktionen zur automatischen Durchführung einer Wahl und ermöglicht die nicht-interaktive Abgabe von Stimmen.

6.1 Installation und Start

Die Installation erfolgt durch Herunterladen und Ausführen der JAR-Datei "Simulation.jar" von

`http://evoting-thesis-lb12.origo.ethz.ch/download`

Da die Simulation eine Instanz eines Bulletin Boards und einer Registrar Authority beinhaltet, wird zudem eine Datenbankverbindung vorausgesetzt. Es gelten dieselben Voraussetzungen zur Installation wie bei der Installation des Bulletin Boards.

Der Start der Simulation kann über die nachfolgenden Parameter gesteuert werden:

```
-beta <size>                    specify the size of beta (min number of
                                replicates for each casted ballot).
-betamax <size>                 specify the limit of beta (value of 0
                                means size of VoterRoll).
-candidateslate <size>          specify the size of the CandidateSlate
                                (number of possible choices used for
                                simulation).
-config <configfile>            read parameters from the specified
                                configfile.
-duplicatevotes <percent>       specify percentage of duplicate votes
                                (0-100).
-fakevotes <percent>            specify percentage of fake votes (0-100).
-help                           print this message
-invalidvotes <percent>         specify percentage of invalid votes
                                (0-100).
-loglevel <level>               use the specified loglevel for the
                                simulation parties (apache log4j levels
                                are valid:
                                FATAL,ERROR,WARN,INFO,DEBUG,TRACE).
-mixingauthorities <number>     specify the number of MixingAuthorities.
-participation <percent>        specify percentage of voter participation
                                (1-100).
-save                           save simulation parameters to config
                                file.
-tallyingauthorities <number>   specify the number of
                                TallyingAuthorities.
-voterroll <size>               specify the size of the VoterRoll (number
                                of registred voters).
```

Keiner der möglichen Start-Parameter ist notwendig zum Starten einer Simulation. Unbekannte Werte werden interaktiv auf der Konsole abgefragt oder der Standardwert übernommen. Weiter nutzt die Simulationskomponente eine Konfigurationsdatei (standardmässig "simulation.properties"), welche es erlaubt, die

Datenbankverbindung (analog Bulletin Board / Registrar Authority, vgl. Abschnitt 5.5) und weitere Einstellungen zu konfigurieren. Der Abschnitt 6.5 befasst sich ausführlich mit den verschiedenen Parametern der Konfigurationsdatei.

6.2 Simulations-Parameter

Das Verhalten der Simulation ist abhängig von den einzelnen Simulations-Parametern. In gewissen Fällen kann dies auch bedeuten, dass die Simulation nicht durchgeführt werden kann (durch die Kombination der Simulations-Parameter können ungültige Wahlsimulationen entstehen). Nachfolgend werden daher die einzelnen Simulations-Parameter und deren Auswirkung beschrieben. Die Parameter werden anhand der Bezeichnung in der Konfigurationsdatei beschrieben. Die Zuteilung von Start-Parameter zu Konfigurations-Parameter sollte jedoch übereinstimmen bzw. eindeutig zuzuweisen sein.

VOTERROLLSIZE: Die Registrar Authority besitzt eine gespeicherte Liste von Wählern mit ihrer Identität (in dieser Arbeit eine Zeichenfolge, z.B. die E-Mail-Adresse) und einem automatisch generierten Geheimnis σ. Dieser Wert ist eine Zufallszahl (standardmässig 1024 Bits). Bei der Publizierung der Voter Roll werden alle σ-Werte der Wähler anhand der ElGamal-Domänen-Parameter p, q und g vom Bulletin Board in ein Element der Gruppe $\mathcal{G}_q$ umgewandelt (mit $g^\sigma \mod p$). Wie viele Werte aus der lokal gespeicherten Liste der Registrar Authority in die Voter Roll integriert werden, wird mit dem Parameter "VOTERROLLSIZE" bestimmt. Sind in der gespeicherten Liste der Registrar Authority nicht genügend Einträge vorhanden, werden automatisch weitere Einträge generiert, resp. Wähler registriert. Ist der Parameter "VOTERROLLSIZE" kleiner als die Anzahl gespeicherter Einträge bei der Registrar Authority werden nur so viele Einträge umgewandelt, wie angegeben wurden.

Grundsätzlich hätte die Partei "Registrar Authority" auch so umgesetzt werden können, dass direkt Elemente aus $\mathcal{G}_q$ bei der Registrierung gespeichert würden. Aufgrund der oben beschriebenen Vorgehensweise stehen die Einträge der Registrar Authority (also die registrierten Wählerdaten) aber für verschiedene Bulletin Board Instanzen, insbesondere Instanzen mit unterschiedlichen p,q und g Werten, zur Verfügung. Damit muss der Datenbestand der Registrar Authority beim Wechsel der ElGamal-Domänen-Parameter nicht neu erstellt werden. Dies ist ein Vorteil für die Simulationen und die Erhebung von Performance-Daten, da es das einfache Vergleichen der Performance der Applikation mit unterschiedlichen ElGamal Domänen-Parametern erlaubt. Bei der Verwendung extrem kleiner Domänen-Parameter steigt bei dieser Form der Umsetzung die Wahrscheinlichkeit, dass durch die Umwandlung der grossen σ-Werte in kleine Gruppen $\mathcal{G}_q$ zwei Wähler mit dem selben Element aus $\mathcal{G}_q$ abstimmen und dadurch eine der beiden Stimmen als Duplikat eliminiert wird. Aus Sicht der Absolventen macht ein Betrieb mit derart kleinen Gruppen allerdings keinen Sinn und das Risiko ist dementsprechend klein. Nichts desto trotz wird beim Starten des Bulletin Boards sichergestellt, dass die Grösse der Voter Roll und die Anzahl der Wahloptionen den Wert q nicht übersteigt.

PARTICIPATION: Dieser Parameter legt fest, wie viele der eingetragenen Wähler aus der Voter Roll abstimmen. Dazu wird die Stimmabgabe für die ersten n Wähler aus der Voter Roll simuliert, bis der Wert

$$n = VOTERROLLSIZE * \frac{PARTICIPATION}{100}$$

erreicht ist. Bei einem Wert von 0 (0% Wählerbeteiligung) werden keine Stimmen abgegeben.

INVALIDVOTES: Dieser Wert hat Einfluss auf die Art der abgegebenen Stimmen. Bei der Simulation der Stimmabgabe wird durch den Zufallsgenerator für jeden eingetragenen Wähler eine Zahl zwischen 0 und 1 generiert und geprüft, ob der Wert kleiner als

$$\frac{INVALIDVOTES}{100}$$

ist. Ist dies der Fall, wird eine Stimme mit einem fehlenden Beweis abgegeben. Die Grundmenge zur Bestimmung der ungültigen Stimmen ist

$$VOTERROLLSIZE \cdot \frac{PARTICIPATION}{100}.$$

Ein Wert von 10 für diesen Parameter bedeutet, dass im Durchschnitt bei 10% der abgegebenen Stimmen ein Beweis fehlt (die Stimmabgabe also ungültig ist).

DUPLICATEVOTES: Analog dem Parameter "INVALIDVOTES" wird durch eine Zufallszahl ermittelt, ob zwei Stimmabgaben durch den Wähler publiziert werden. Falls ja, wird für den aktuellen Wähler zuerst eine Stimme für die zweite Wahloption abgegeben und danach eine Stimme für die erste. Durch die in dieser Arbeit umgesetzte Richtlinie für Duplikate ("last-vote-wins"), wird die zuerst abgegebene Stimme für die zweite Wahloption als Duplikat markiert. Eine Stimme kann nur ein Duplikat werden, wenn sie nicht ungültig ist. Die Grundmenge für Duplikate ist also

$$VOTERROLLSIZE \cdot \frac{PARTICIPATION}{100} \cdot \frac{100 - INVALIDVOTES}{100}.$$

Das bedeutet, wenn der Parameter "INVALIDVOTES" 10% beträgt und der Parameter "DUPLICATEVOTES" 20%, werden 20% von 90% abgegebenen Stimmen doppelt generiert (das ergibt 18% der gesamten, abgegebenen Stimmen).

FAKEVOTES: Mit diesem Parameter wird festgelegt, wie viele gefälschte Stimmen abgegeben werden sollen. Die Simulation einer gefälschten Stimme ist relativ aufwendig, da sichergestellt werden muss, dass das verwendete (gefälschte) σ nicht in der Voter Roll (von einem anderen Wähler) benutzt wird. Die Umsetzung in dieser Arbeit sieht vor, dass bei der Publizierung der Voter Roll durch die Registrar Authority entsprechend dem Prozent-Anteil gefälschter Stimmen Einträge nicht in die Voter Roll integriert werden.

Dies führt zum Verhalten, dass Wähler eine Stimme abgeben, die nicht in der Voter Roll enthalten sind (und somit gefälschte Stimmen entstehen). Diese Art der Umsetzung verhindert jedoch die Durchführung einer Simulation mit 100% Anteil gefälschter Stimmen. Das ist so, weil bei der Stimmabgabe das Anonymitätsset nicht gebildet werden kann, wenn nicht genügend Einträge in der Voter Roll existieren.

Die Grundmenge des Parameters "FAKEVOTES" ist der Wert des Parameters "VOTERROLLSIZE". Bei 10% Anteil gefälschter Stimmen wird also im Durchschnitt jeder 10. Eintrag bei der Publizierung der Voter Roll nicht berücksichtigt.

BETA / BETAMAX: Die Parameter "BETA" und "BETAMAX" definieren die Grösse des Anonymitätssets. Das bedeutet, für jede abgegebene Stimme wird nach der Phase "Duplicate Vote Elimination" zwischen "BETA" und "BETAMAX" Stimmen repliziert. Wird "BETAMAX" auf 0 gesetzt bedeutet dies, es werden zwischen "BETA" und der Anzahl Einträge in der Voter Roll Stimmen repliziert. Beim Statuswechsel des Bulletin Boards von "Setup" nach "Vote Casting" wird geprüft, ob der Wert "BETAMAX" grösser ist als "BETA" (oder 0) und ob "BETAMAX" kleiner oder gleich der Grösse der Voter Roll ist.

CANDIDATESLATESIZE: Mit diesem Parameter wird die Anzahl möglicher Wahloptionen (Grösse der "Candidate Slate") bestimmt. Dieser Wert hat einen Einfluss auf die Generierung der Stimmen, da bei der Stimmabgabe ein Proof of Validity erzeugt werden muss, welcher von der Anzahl Wahloptionen abhängig ist. Es ist offensichtlich, dass die minimale Anzahl 2 beträgt (ansonsten ist keine Auswahl möglich). Auch diese Bedingung wird beim Statusübergang von "Setup" nach "Vote Casting" vom Bulletin Board überprüft.

NUMTALLYINGAUTHORITIES: Dieser Parameter definiert die Anzahl teilnehmender Tallying Authorites. Ein Wert kleiner als eins ist nicht zulässig. Der Parameter wirkt sich auf die Ausführungszeit der Phasen "Duplicate Vote Elimination", "Fake Vote Elimination" und "Tallying" aus.

NUMIXINGRUNS: Dieser Parameter legt fest, wie viele Mixing-Durchläufe (also Mixing Authorities) in der simulierten Wahl existieren. Ein Wert kleiner als eins ist nicht zulässig, wobei wie bereits erwähnt bei allen Applikationsrollen davon ausgegangen wird, dass die Parteien vertrauenswürdig sind.

Mit Hilfe dieser Simulations-Parameter lassen sich unterschiedliche Szenarien simulieren und auswerten. Die Szenarien reichen dabei von Simulationen mit unterschiedlicher Wahlbeteiligung über eine variierende Anzahl teilnehmender Mixing bzw. Tallying Authorities bis hin zu einem quadratischen Laufzeitverhalten durch Anpassung des Sicherheits-Parameters "BETA" (in Kombination mit "BETAMAX").

6.3 Der Ablauf einer Simulation

Damit eine Simulation ausgeführt werden kann, werden im ersten Schritt die Simulations-Parameter bestimmt. Dies geschieht durch das Auslesen der Simulations-Konfigurationsdatei, resp. durch die übergebenen Start-Parameter. Falls der Wert weder in der Konfigurationsdatei noch als Parameter übergeben wurde, wird interaktiv danach gefragt (über die Konsole). Falls ein Wert sowohl in der Konfigurationsdatei enthalten ist wie auch als Start-Parameter übergeben wurde, hat der Wert aus dem Start-Parameter Vorrang.

Exemplarisch simulieren wir eine Wahl von 100 registrierten Wählern mit fünf Wahloptionen, 100% Wahlbeteiligung, 0% ungültigen Stimmen ("INVALIDVOTES"), 0% mehrfachen Stimmabgaben ("DUPLICATEVOTES") und 0% gefälschten Stimmen ("FAKEVOTES"). Wir setzen als Sicherheits-Parameter "BETA" einen Wert von 5 und limitieren ihn auf 20 ("BETAMAX"). Für jeden Wähler werden also zwischen 5 und 20 Stimmen repliziert. Wir simulieren zwei Tallying Authorities und drei Mixing Authorities.

```
java -jar Simulation.jar -save
...
*** specifiy size of the VoterRoll (number of voters used for this simulation): 100
*** specifiy size of the CandidateSlate (number of choices used for this simulation): 5
*** specifiy size of beta (min number of replicates for each casted ballot): 5
*** specifiy the limit of beta (value 0 means limit = size of VoterRoll): 20
*** specifiy the number of TallyingAuthorities: 2
*** specifiy the number of MixingAuthorities: 3
*** specify percentage of voter participation (1-100): 100
*** specify percentage of invalid votes (1-100): 0
*** specify percentage of duplicate votes (1-100): 0
*** specify percentage of fake votes (1-100): 0
19457 [main] INFO Simulation  - overview of the simulation parameters:
19457 [main] INFO Simulation  - VOTERROLLSIZE: 100
19457 [main] INFO Simulation  - CANDIDATESLATESIZE: 5
19457 [main] INFO Simulation  - BETA: 5
19457 [main] INFO Simulation  - BETAMAX: 20
19457 [main] INFO Simulation  - NUMTALLYINGAUTHORITIES: 2
19457 [main] INFO Simulation  - NUMIXINGRUNS: 3
19457 [main] INFO Simulation  - PARTICIPATION: 100
19457 [main] INFO Simulation  - INVALIDVOTES: 0
19457 [main] INFO Simulation  - DUPLICATEVOTES: 0
19457 [main] INFO Simulation  - FAKEVOTES: 0
...
```

Durch den erstmaligen Start der Simulation (ohne existierende Konfigurationsdatei) fragt die Simulation alle Simulations-Parameter ab (es wurden keine Start-Parameter übergeben). Mittels Angabe der Option "save" werden diese Simulations-Parameter in die neu erstellte Konfigurationsdatei "simulation.properties" übernommen. Bei künftigen Starts der Simulation werden nun die Parameter aus der Konfigurationsdatei gelesen und es werden keine Fragen mehr an den Benutzer gestellt (sofern der Aufruf aus demselben Verzeichnis geschieht).

Wir wollen mehrere Simulationsdurchläufe starten und mit den Werten der ungültigen, mehrfachen und gefälschten Stimmen variieren. Dazu werden die drei Parameter "INVALIDVOTES", "DUPLICATEVOTES" und "FAKEVOTES" aus der im obigen Schritt erstellten Konfigurationsdatei entfernt. Der erneute Aufruf der Simulation fragt nur noch diese drei Werte ab:

```
java -jar Simulation.jar
...
*** specify percentage of invalid votes (1-100): 10
*** specify percentage of duplicate votes (1-100): 20
*** specify percentage of fake votes (1-100): 10
125446 [main] INFO Simulation  - overview of the simulation parameters:
125446 [main] INFO Simulation  - VOTERROLLSIZE: 100
125446 [main] INFO Simulation  - CANDIDATESLATESIZE: 5
125447 [main] INFO Simulation  - BETA: 5
```

```
125447 [main] INFO Simulation  - BETAMAX: 20
125447 [main] INFO Simulation  - NUMTALLYINGAUTHORITIES: 2
125447 [main] INFO Simulation  - NUMIXINGRUNS: 3
125447 [main] INFO Simulation  - PARTICIPATION: 100
125447 [main] INFO Simulation  - INVALIDVOTES: 10
125447 [main] INFO Simulation  - DUPLICATEVOTES: 20
125447 [main] INFO Simulation  - FAKEVOTES: 10
...
```

Da der Start-Parameter "save" nicht angegeben wurde, werden diese Werte nicht in die Konfigurationsdatei gespeichert und bei der nächsten Ausführung wieder abgefragt.

6.3.1 Ablauf bei der Ausführung der Simulation

Nachdem die Benutzung der Simulationskomponente aufgezeigt wurde, soll detailliert beschrieben werden, was die Simulation nach der Abfrage der Simulations-Parameter ausführt.

Im ersten Schritt werden die drei RMI-Server Komponenten Bulletin Board, Board Authority und Registrar Authority generiert und gestartet. Alle Komponenten werden mit dem entsprechend gesetzten Log Level des Simulations-Parameters und der Log Datei "simulation_board.log" bzw. "simulation_boardauthority.log" oder "simulation_registrar.log" konfiguriert. Die RMI-Services werden auf dem lokalen Rechner mit den Standard-Ports (siehe Abschnitt 5.5) initiiert. Nach diesem Schritt steht ein Bulletin Board mit den Werten der Simulations-Parameter im Status "Setup" zur Verfügung, sowie eine Registrar Authority und eine Board Authority.

Als nächstes werden nun von den Tallying Authorites die Teilschlüssel und Teilverblindungswerte inkl. dem entsprechenden Commitment generiert. Dabei werden entsprechend der Anzahl Tallying Authorities Instanzen gestartet und für jede Tallying Authority der Arbeitsschritt "init" ausgeführt. Da jede Tallying Authority eine Konfigurationsdatei benötigt, wird automatisiert pro Tallying Authority eine Konfigurationsdatei namens "simulation_taN.properties" erstellt, wobei N für die Instanznummer steht (beginnend mit 0). Sobald die letzte Tallying Authority ihre Teilresultate auf dem Bulletin Board gespeichert hat, berechnet die Board Authority den gesamten öffentlichen Schlüssel und publiziert diesen auf dem Bulletin Board.

In einem weiteren Schritt wird der Abstimmungsname fix auf "simulated vote (every valid vote casts for 'option 1')" gesetzt und entsprechend der Anzahl Wahloptionen n Abstimmungsmöglichkeiten mit der Beschriftung "option x" ($x \in \{1, \ldots, n\}$) generiert und auf dem Bulletin Board abgelegt. Zuletzt wird die Voter Roll durch die Registrar Authority auf dem Bulletin Board publiziert. Die Einrichtung des Bulletin Boards wird abgeschlossen durch einen simulierten Administrator-Zugriff, mit welchem der Status des Boards auf "Vote Casting" gesetzt wird.

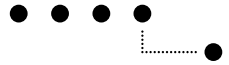
Es folgt die Simulation der Stimmabgaben durch die fiktiven Wähler. Hierbei wird durch alle Einträge der Registrar Authority iteriert und für jeden Wähler eine Stimme abgegeben (abhängig von den Parametern "PARTICIPATION","INVALIDVOTES" und "DUPLICATEVOTES").

Danach folgen die Phasen "Invalid Vote Elimination" (Zuständigkeit: Board Authority), "Duplicate Vote Elimination" (Tallying Authorities und Board Authority im Verbund), "Mixing Ready" und "Mixing In Progress" (Mixing Authorities), "Fake Vote Elimination" und "Tallying" (erneut Tallying Authorities und Board Authority im Verbund).

Zum Schluss liefert die Simulation eine Übersicht über die Wahl (abgegebene Stimmen, Anzahl ungültige, mehrfache und gefälschte Stimmen). Zudem werden eine Reihe von Performance-Werten im CSV-Format ausgegeben. Der Aufbau dieser Performance-Ausgabe wird im Kapitel 7 beschrieben.

6.4 Interpretation der Simulationsausgaben

Die Applikationsteile Bulletin Board, Board Authority und Registrar Authority verwenden bei der simulierten Wahl, wie bereits erwähnt, jeweils eine eigene Logdatei ("simulation_board.log", "simulation_boardauthority.log"



und “simulation_registrar.log”). Daraus können detaillierte Ergebnisse der einzelnen Entitäten entnommen werden (entsprechend dem gesetzten / parametrisierten Log Level beim Simulationsstart). Die Simulation selbst zeichnet ebenfalls wichtige Schritte auf, wie z.B. Start und Ende jeder Phase. In der ersten Spalte der Log Meldungen ist dazu die Anzahl Millisekunden seit Programmstart enthalten. Meldungen der Simulation zu Start bzw. Ende einer Phase beginnen jeweils mit dem Muster “PERF”.

6.5 Konfiguration

In der Konfigurationsdatei (standardmässig “simulation.properties”) werden grundsätzlich zwei Arten von Parametern gespeichert:

- Programm-Parameter
- Simulations-Parameter

Die Simulations-Parameter wurden bereits ausführlich diskutiert. Die Programm-Parameter dienen zur Konfiguration der Datenbankverbindung, des ElGamal Setups und des Log Levels. Sie besitzen im Gegensatz zu den Simulations-Parametern Standardwerte, welche beim Start ohne Konfigurationsdatei gesetzt werden.

6.5.1 Programm-Parameter

- G
  Der Wert des ElGamal-Domänen-Parameters g. Bei g muss es sich um einen gültigen Generator der Gruppe handeln (siehe Abschnitt 4.12.1).
  Standardwert: (Generator der Gruppe q)
- JPA_DRIVER
  Dieser Wert konfiguriert zusammen mit dem Parameter “JPA_URL” die Datenbankverbindung. Mittels Angabe des Treibers wird das benutzte Eclipselink (JPA) Framework konfiguriert.
  Standardwert: com.mysql.jdbc.Driver
- JPA_PASSWD
  Das Passwort für die Verbindung zur Datenbank.
  Standardwert: -
- JPA_URL
  Dieser Wert konfiguriert zusammen mit dem Parameter “JPA_DRIVER” die Datenbankverbindung. Mittels Angabe der URL wird der Datenbank-Rechner, Port und Datenbank-Name spezifiziert. Der Standardwert konfiguriert eine MySQL-Datenbank auf dem lokalen Rechner (TCP-Port 3306) und der Datenbank “Test”.
  Standardwert: jdbc:mysql://localhost:3306/test
- JPA_USER
  Der Benutzername für die Verbindung zur Datenbank.
  Standardwert: root
- LOGLEVEL
  Dieser Wert definiert den Log Level der automatisiert gestarteten Applikationsteile (Bulletin Board, Registrar, Board Authority, ...).
  Standardwert: INFO
- P
  Der Wert des ElGamal-Domänen-Parameters p. Der Wert p muss eine Primzahl sein und aus $p = k*q+1$ gebildet sein (siehe Abschnitt 4.12.1).
  Standardwert: (1024-Bit Wert)
- Q
  Der Wert des ElGamal-Domänen-Parameters q. Der Wert q muss eine Primzahl sein (siehe Abschnitt 4.12.1).
  Standardwert: (160-Bit Wert)

6.5.2 Simulations-Parameter

- BETA
  Der Wert für den Sicherheits-Parameter "beta".
- BETAMAX
  Dieser Wert konfiguriert das Limit für "beta". Der Wert 0 bedeutet, dass "BETAMAX" gleich der Grösse der Voter Roll ist.
- CANDIDATESLATESIZE
  Dieser Wert konfiguriert, wie viele Wahloptionen bei der Wahl zur Verfügung stehen.
- DUPLICATEVOTES
  Dieser Wert konfiguriert den Anteil an Duplikaten, der simuliert werden sollen. Ein Wert von 10% bedeutet, dass 10% der Stimmabgaben (abzgl. der ungültigen Stimmen) doppelt abgegeben werden.
- FAKEVOTES
  Dieser Wert konfiguriert das Verhalten für die Abgabe von gefälschten Stimmen. Analog dem Wert "DUPLICATEVOTES" handelt es sich um eine Prozentzahl.
- INVALIDVOTES
  Legt den Anteil ungültiger Stimmen (ungültige oder fehlende Zero-Knowledge Proofs) in Prozent fest.
- NUMIXINGRUNS
  Dieser Wert legt fest, wie viele Mixing Authorities an der simulierten Wahl teilnehmen.
- NUMTALLYINGAUTHORITIES
  Mit diesem Wert wird festgelegt, wie viele Tallying Authorities an der simulierten Wahl teilnehmen.
- PARTICIPATION
  Mit diesem Wert kann eine Wahlbeteiligung festgesetzt werden. Ein Wert von 80 bedeutet, 20% der Wähler aus der Voter Roll stimmen nicht ab.
- VOTERROLLSIZE
  Dieser Wert legt die Grösse der Voter Roll fest, also wie viele Wähler abstimmen dürfen.

6.5.3 Beispiel

Wird die nachfolgende Konfigurationsdatei im Verzeichnis abgelegt, in welchem der Befehl "java -jar" (mit Angabe der JAR-Datei) gestartet wird und die Datei mittels Parameter "-config" übergeben, so wird bei jedem Simulationsdurchlauf nach den Werten für die Grösse der Voter Roll, "BETA" und "BETAMAX" gefragt (da diese Simulationswerte in der Konfigurationsdatei nicht enthalten sind).

```
CANDIDATESLATESIZE=2
DUPLICATEVOTES=10
FAKEVOTES=10
G=774152683191151438768283121789...
INVALIDVOTES=10
JPA_DRIVER=com.mysql.jdbc.Driver
JPA_PASSWD=
JPA_URL=jdbc:mysql://localhost:3306/test
JPA_USER=root
LOGLEVEL=INFO
NUMIXINGRUNS=2
NUMTALLYINGAUTHORITIES=2
P=985613645828315525065325184 2...
PARTICIPATION=100
Q=10188230187988047...
```

Sofern das aktuelle Verzeichnis sowohl die Konfigurationsdatei ("mysimulation.properties") als auch die JAR-Datei enthält, ist der Aufruf der Simulation wie folgt:

```
java -jar Simulation.jar -config mysimulation.properties
```

7 Performance

Dieses Kapitel soll die Ergebnisse der Performance-Tests bzw. der Simulation von verschiedenen Szenarien aufzeigen. Einerseits soll bestätigt werden, dass die Implementation das in der Theorie entwickelte, lineare Laufzeitverhalten gemäss dem Protokoll SHKS11 aufweist (im Unterschied zum quadratischen Laufzeitverhalten von JCJ05). Andererseits sollen die ermittelten Daten auch Rückschlüsse auf die maximale Anzahl von Stimmen geben, welche durch die Applikation bearbeitet werden kann.

7.1 Anzahl Einträge auf dem Bulletin Board

Bereits bei der Entwicklung der Applikation und vor allem bei der Simulation von verschiedenen Szenarien ist die massive Vervielfachung der abgegeben Stimmen aufgefallen. Es ist anhand der Parameter "BETA" und "BETAMAX", sowie der Anzahl Tallying und Mixing Authorites möglich, diese Vervielfachung zu steuern. Die Anzahl der Einträge auf dem Board kann dabei das 100-fache der abgegebenen Stimmen erreichen. Gründe dafür sind die Replizierung der Stimmen, das Mixen der replizierten Stimmen, die Einträge jeder Tallying Authority pro Stimme für die Teilverblindung, Teilentschlüsselung und das Auszählen.

Als Beispiel diene das Szenario von 1'000 registrierten Wählern (100%ige Wahlbeteiligung, 10% ungültige, 20% doppelte und 10% gefälschte Stimmen), zwei Tallying Authorites, drei Mixing Authorities und die Werte 5 für "BETA" und 20 für "BETAMAX". Das Wachstum der Einträge zeigt die nachfolgende Auflistung:

- 1'180 Einträge nach Stimmabgabe (die Einträge durch das Setup werden ignoriert). Diese Zahl setzt sich zusammen aus 100 ungültigen Stimmen, 180 Duplikaten (=360 Stimmen) und 720 anderen (möglicherweise gefälschten) Stimmen.
- 1'280 Einträge nach der Phase "Invalid Vote Elimination": Es kommen 100 Einträge (Markierungen) hinzu.
- 6'860 Einträge nach der Phase "Duplicate Vote Elimination": Es kommt pro Tallying Authority für jede Stimme je ein Eintrag für Teilverblindung und Teilentschlüsselung hinzu. Dazu kommt das Gesamtresultat der Verblindung / Entschlüsselung pro Stimme von der Board Authority.
- 15'860 Einträge nach der Replizierung der Stimmen: Durch "BETA" und "BETAMAX" kommen durchschnittlich für jede, noch gültige Stimme 12.5 neue Einträge hinzu (9'000 replizierte Stimmen).
- 42'860 Einträge nach dem Mixing: Es kommt pro Mixing Authority die Anzahl replizierter Stimmen hinzu (jede Mixing Authority legt erneut 9'000 Stimmen ab).
- 96'860 Einträge nach der Phase "Fake Vote Elimination": Für die 9'000 durchmischten Stimmen werden erneut von jeder Tallying Authority und von der Board Authority je zwei Einträge für die Verblindung / Entschlüsselung abgelegt (plus 54'000 Einträge).
- Durch das Auszählen der verbleibenden, gültigen Stimmen (in diesem Szenario ca. 600–700 Stimmen) erreicht das Board somit ca. 100'000 Einträge.

Die Messdaten von über 50 durchgeführten Simulationen mit 10 bis 1'000 registrierten Wählern zeigen im Durchschnitt eine Multiplikation der ursprünglich abgegebenen Stimmen von 50. Das bedeutet im Durchschnitt wurden die abgegeben Stimmen um das 50-fache vervielfacht (bezogen auf die Anzahl Board Einträge).

7.2 Messverfahren

Als Messverfahren wurde die in Kapitel 6 vorgestellte Simulation mit entsprechend angepassten Simulations-Parametern verwendet. Bei der für die Messung eingesetzten Infrastruktur handelt es sich um insgesamt drei unterschiedliche Endbenutzersysteme (siehe Anhang F).

Die Simulationskomponente speichert bei der Durchführung der Simulation die von der Java Runtime ermittelte Anzahl Millisekunden (seit 1. Januar 1970) für den Start- und End-Zeitpunkt jeder Phase. Durch Subtraktion der Werte für Ende und Start der Phase wird die benötigte Zeitdauer ermittelt. Die letzte Ausgabe nach dem Simulationsdurchlauf liefert neben den Simulations-Parametern weitere Angaben für die Auswertung der Daten im CSV-Format:

```
PERF: Performance data 10321;1040;6170;2544;40593;24783;3611;50;5;3;2;5;20;90;10;20;0
```

Diese Werte können für die weitere Analyse in andere Programmen oder Datenbanken importiert werden. Die Bedeutung der einzelnen Spalten (von links nach rechts) sind:

1. Anzahl Millisekunden für die simulierten Stimmabgaben (timeVoteCasting)
2. Anzahl Millisekunden für Phase "Invalid Vote Elimination" (timeInvalidVote)
3. Anzahl Millisekunden für Phase "Duplicate Vote Elimination" (timeDuplicateVote)
4. Anzahl Millisekunden für Phase "Ballot Replication" (timeBallotReplication)
5. Anzahl Millisekunden für Phasen "Mixing Ready" und "Mixing In Progress" (timeMixing)
6. Anzahl Millisekunden für Phase "Fake Vote Elimination" (timeFakeVote)
7. Anzahl Millisekunden für Phase "Tallying" (timeTallying)
8. Grösse der Voter Roll (sizeVoterRoll)
9. Grösse der Candidate Slate (sizeCandidateSlate)
10. Anzahl Mixing Authorities (noMixingAuth)
11. Anzahl Tallying Authorities (noTallyingAuth)
12. Wert von "BETA"
13. Wert von "BETAMAX"
14. Wahlbeteiligung in Prozent (participation)
15. Prozent ungültiger Stimmen (invalidVotes)
16. Prozent doppelter Stimmabgaben (dulicateVotes)
17. Prozent gefälschter Stimmabgaben (fakeVotes)

Die Summe der Anzahl Millisekunden der Spalten 2–6 repräsentiert dabei die benötigte Zeit für die "Vote Authorization" gemäss Protokoll. Diese Zeitangaben werden jedoch durch diverse Faktoren beeinflusst, z.B. durch die Performance der Java Virtual Machine (unterschiedliche Java Runtime Versionen haben möglicherweise unterschiedliche Ausführungszeiten). Relevant ist auch der Zeitbedarf für die Interaktion mit der Datenbank, resp. die verwendete Kombination von Datenbank und Framework (in unserem Fall MySQL und Eclipselink). Entscheidend sind natürlich insbesondere die verwendete Hardwareplattform und allenfalls parallel laufende Prozesse, welche ebenfalls Rechenleistung benötigen.

Neben den genannten Ungenauigkeiten bei der Ermittlung der Messwerte ist besonders die Art der Durchführung zu beachten: Bei der Simulation werden standardmässig alle Komponenten auf einem Rechner ausgeführt, was nicht einem realen Szenario entspricht. Ein weiterer Unterschied ist die Durchführung der Berechnungen der Tallying Authorities: Diese Arbeitsschritte sind parallel durchführbar (im Unterschied zum Mixing), die Simulationskomponente führt die Berechnungen der einzelnen Tallying Authorities jedoch sequentiell aus. Dies wurde aufgrund der höheren Komplexität durch Multi-Threading Applikationen und der Fehleranfälligkeit durch Race Conditions so umgesetzt. Es sei an dieser Stelle angemerkt, dass vordergründig die Umsetzung der Spezifikation des Protokolls im Zentrum der Implementation stand. Es ist daher davon auszugehen, dass diverse Programmteile optimiert werden können hinsichtlich ihrer Performance.

7.3 Laufzeitverhalten

Wie bereits mehrfach erwähnt, besitzt das Protokoll JCJ05 ein quadratisches Laufzeitverhalten bezogen auf die Anzahl abgegebener Stimmen, resp. der registrierten Wähler. Eines der Ziele dieser Arbeit war es, das lineare Laufzeitverhalten der Variante SHKS11 praktisch aufzeigen zu können.

Für die Analyse des Laufzeitverhaltens wurden mit unveränderten Werten der Parameter "BETA", "BETAMAX", "NUMTALLYINGAUTHORITIES", "NUMIXINGRUNS", "INVALIDVOTES", "DUPLICATEVOTES" und "FAKEVOTES" Simulationsdurchläufe mit einer unterschiedlichen Anzahl von registrierten Wählern (50–1'000) durchgeführt. In der Abbildung 7.1 dargestellt sind die durchschnittlichen Ausführungszeiten pro Wählergruppe, wie auch der theoretisch errechnete, lineare Verlauf. Für die Berechnung des theoretischen Laufzeitverhaltens dienen die Messdaten von 50 Wählern auf dem "Desktop PC"-System (nach Anhang F) als Grundwert, welche entsprechend hoch gerechnet werden.

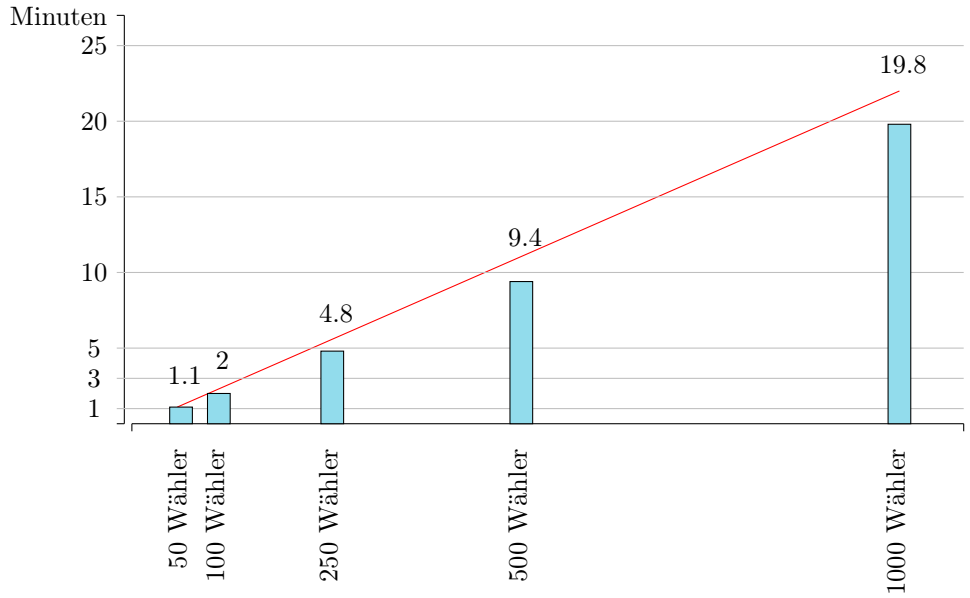


Abbildung 7.1: Laufzeitverhalten verschiedener Szenarien

Das theoretisch errechnete Laufzeitverhalten entspricht der roten Linie, die Messresultate sind durch blaue Säulen dargestellt. Für alle Messungen gilt der Durchschnittswert aus mindestens drei Messungen. Aufgrund dieser Messungen zeigt sich das lineare Laufzeitverhalten der Applikation.

Die Erfahrungen aus den Simulationen zeigen, dass vor allem die Wahl des Parameters "BETA" bzw. dessen Limitierung "BETAMAX" ausschlaggebend für die Performance ist. Obwohl das Protokoll ein Anonymitätsset mit einer maximalen Grösse der Voter Roll vorsieht, ist es aus Sicht der Absolventen sinnvoll den Wert stärker zu limitieren. Dies verbessert die Verarbeitungsgeschwindigkeit massiv.

Die Applikation bzw. die Java Virtual Machine sollte bei der Simulation bzw. der Durchführung von Wahlen mit mehr als 100'000 Datenbankobjekten (ca. ab 1'000 Wähler mit "BETA" / "BETAMAX" von 5 bzw. 20) mit einem Parameter gestartet werden. Es ist möglich, dass aufgrund der massiven Anzahl Objekte die Java Virtual Machine den Fehler

```
Exception in thread "main" java.lang.OutOfMemoryError: GC overhead limit exceeded
```

generiert. Dieser Fehler wird auf dem Technetwork von Oracle[1] beschrieben und sagt aus, dass zu viel CPU-Zeit für die Garbage Collection verwendet wurde (und somit zu wenig CPU-Zeit für die Ausführung des Programms). Dieses Verhalten kann bei grösseren Abstimmungen mit folgender Option deaktiviert werden:

```
-XX:-UseGCOverheadLimit
```

[1] `http://www.oracle.com/technetwork/java/javase/gc-tuning-6-140523.html`, Abschnitt "Excessive GC Time and OutOfMemoryError"

8 Fazit

In diesem letzten Kapitel wollen wir die Erkenntnisse aus dieser Arbeit zusammenfassen, Bilanz ziehen und abschliessend die Arbeit als solche reflektieren.

Die Endversion der Applikation ist aufgeteilt in 20 verschiedene Packages und beinhaltet insgesamt 9'754 Zeilen Java Code (ohne Kommentare). Abbildung 8.1 zeigt die Entwicklung der Workitems auf der Projektplattform Origo[1]. Als Workitem werden sowohl Projektversionen (Commits), Aufgaben (Issues), Blogs (Besprechungsnotizen) als auch offizielle Releases bezeichnet. Da die Origo Plattform an Neujahr einen mehrtätigen Ausfall verzeichnete, fehlen für diesen Zeitraum die entsprechenden Daten in der Grafik.

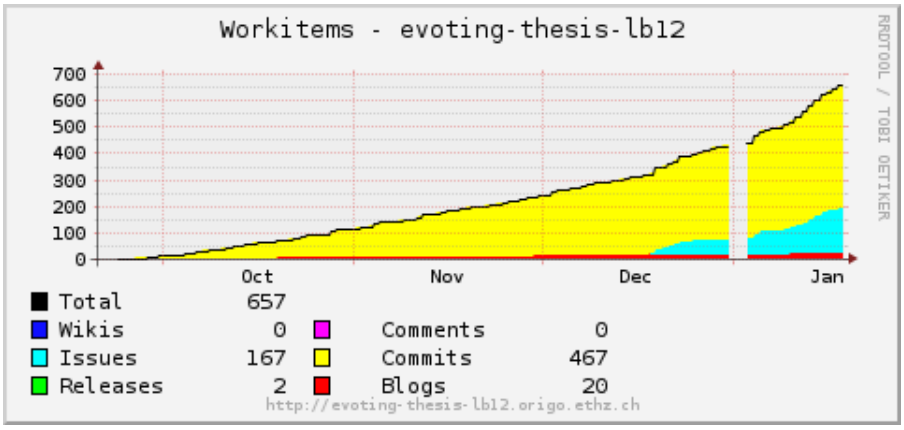


Abbildung 8.1: Workitems Statistik von evoting-thesis-lb12.origo.ethz.ch

8.1 Ergebnisse

Folgend ein Vergleich der im Abschnitt 1.1 definierten Projektziele:

- Eine Abstimmung kann nach Ansicht der Absolventen gemäss Spezifikation von SHKS11 durchgeführt werden. Die Überprüfung von Zero-Knowledge Proofs ist gewährleistet und wird sowohl in der Vote Authorization als auch vom Verifier angewendet. Es wurden zwei grafische Benutzerumgebungen (Wähler und Administrator) realisiert sowie eine Logging Komponente, welche detaillierte Informationen über den Abstimmungsverlauf gibt. Die Musskriterien wurden aus Sicht der Absolventen erfüllt.

- Es ist möglich eine Wahl anhand entsprechender Parameter zu simulieren (siehe Kapitel 6) und komplett automatisiert ausführen zu lassen. Das Wahlresultat beinhaltet stets auch die Anzahl ungültiger Stimmen, eine Erkennung in Form der Vote Authorization ist fest implementiert. Im Bereich der Performance konnten gewisse Aussagen gemacht werden (siehe Kapitel 7). Durch den Einsatz von Java RMI können die verschiedenen Parteien unabhängig voneinander auf verschiedenen Systemen operieren. Eine Aufteilung in einen Client- und Serverteil ist daher gewährleistet. Für die Tallying Authorities wurde ein Distributed Verfahren entwickelt, welches nicht nur bei der Entschlüsselung der ElGamal Tupel eingesetzt wurde, sondern auch für die Berechnung des öffentlichen Schlüssels und des Verblindungswertes. Die Sollkriterien sind daher aus Sicht der Studenten ebenfalls erfüllt.

- Auf einen Einsatz von Threshold Decryption wurde verzichtet, da dies einen wesentlichen Mehraufwand bedingt hätte. Hingegen wurde die Unterstützung beliebig vieler Wahloptionen (anstelle von nur "ja" oder "nein") implementiert, was unter anderem auch vom Proof of Validity abhängig war. Eine Sicherstellung

[1] `http://evoting-thesis-lb12.origo.ethz.ch`

des “append-only” Modus vom Bulletin Board über entsprechende Signaturen oder Hashes findet nicht statt. Auf eine Authentifizierung der verschiedenen Parteien (z.B. der Mixer) wurde ebenfalls verzichtet. Die Kannkriterien wurden aus Sicht der Absolventen teilweise erfüllt.

8.2 Vergleich Planung SOLL/IST

Erwartungsgemäss kam es gegenüber dem ursprünglichen Projektplan (siehe Abschnitt 1.5) zu Abweichungen:

- Bei der Implementation des Proof of Validity zeigte sich, dass Lücken in den vorgängig erarbeiteten, kryptografischen Grundlagen bestanden, welche zuerst aufgearbeitet werden mussten. Um eine objektorientierte Umsetzung zu ermöglichen, wurden die Zero-Knowledge Proofs schliesslich soweit wie möglich abstrahiert (siehe Abschnitt 4.12.2).
- Im Bereich der Vote Authorization stellten insbesondere die verteilten Berechnungen durch die Tallying Authorities eine Herausforderung dar. Da das Bulletin Board gemäss Auftrag so wenig Funktionalität wie möglich beinhalten sollte, wurde mit den Betreuern vereinbart, die unabhängige Instanz “BoardAuthority” einzuführen, welche die entsprechenden Zwischenresultate der einzelnen Parteien zusammenrechnet.
- Die Abklärungen bezüglich Verificatum (siehe Anhang D) nahmen wesentlich mehr Zeit in Anspruch als geplant. Sie führten unter anderem auch dazu, dass lange Zeit nicht bekannt war, ob Verificatum verteilte Berechnungen im Treshold Modus übernehmen sollte oder, ob sich der Einsatz der Library nur (wie ursprünglich vorgesehen) auf das Mixing beschränken sollte. Mit der Entscheidung, Verificatum nicht im Rahmen dieses Projekts zu implementieren, wurde bestimmt, dass ein Distributed Verfahren und eine eigene Reencryption Mix-Network Implementation zum Einsatz kommt.
- Entgegen der ursprünglichen Planung und auf Empfehlung des Experten wurde die Entwicklung der grafischen Benutzeroberflächen für Wähler und Administrator bereits in der KW43 in Angriff genommen anstatt wie geplant in der KW49, was unter anderem eine anschaulichere Demonstration der Applikation ermöglichte.

In Abbildung 8.2 ist die Planung aus Abschnitt 1.5 mit den tatsächlich verwendeten Daten aktualisiert worden, um den SOLL-/IST-Vergleich darzustellen. Die erste Zeile steht jeweils für den SOLL-Stand (blau), die zweite für die effektive Umsetzung. Ein grüner Balken bedeutet, Planung und Umsetzung waren übereinstimmend bezüglich Ausführungszeit und Dauer. Ein roter Balken deutet an, dass die Planung hinsichtlich der effektiven Dauer nicht übereinstimmte und entweder mehr oder weniger Zeit beansprucht wurde, oder der entsprechende Vorgang verschoben wurde.

Grundsätzlich konnten alle geplanten Bereiche bearbeitet und umgesetzt werden. Zu erwähnen gilt es die wesentlich verlängerte Analyse von Verificatum, welche jedoch wichtige Erkenntnisse brachte, die im Anhang D ausgewiesen werden. Neben dieser Verzögerung war die Umsetzung der Vote Authorization aufwändiger als geplant, was insbesondere auf Kosten des JUnit Testings ging.

8.3 Weiterentwicklung

Es ist nicht ausgeschlossen, dass die vorliegende Arbeit in anderen Projekten zur Anwendung kommt. Die nachfolgenden Punkte wiederspiegeln die aktuelle Meinung der Studenten im Bezug auf eine Weiterentwicklung der Arbeit.

Registrierung: Das Registrierungsprozedere wurde stark vereinfacht um eine simulierte Stimmabgabe zu ermöglichen. Die Registrierung eines Wählers über den von JCJ05 und SHKS11 geforderten “Untappable Channel” stellt zusätzliche Anforderungen und muss als separates Problem betrachtet werden.

append-only Eigenschaft des Boards Die Applikation selbst bietet keine Möglichkeit, um bestehende Einträge vom Bulletin Board zu entfernen oder zu verändern. Über einen direkten Datenbankzugriff ist dies jedoch nach wie vor möglich. Um ein Löschen / Mutieren von Einträgen zu verhindern, könnte beispielsweise ein Hash-Wert über die existierenden Einträge generiert werden. Wird ein Eintrag gelöscht, wäre dies relativ einfach feststellbar, aufgrund des fehlerhaften Hash-Wertes (iteriern über alle Einträge). Von dieser Lösung ausgenommen ist allerdings der letzte Eintrag der Datenbank. Das Löschen dieses letzten Eintrags kann ermittelt werden, wenn jeweils eine Quittung bei der Erstellung ausgegeben wird.

Abbildung 8.2: Vergleich Projektplanung Planung SOLL/IST

Verifiable Reencryption Mix-Network: Die implementierte Version des Reencryption Mix-Networks ist zu grossen Teilen nicht überprüfbar, da sie keine entsprechenden Beweise liefert. Wie in Anhang D erwähnt, musste aus zeitlichen Gründen auf eine Integration von Verificatum verzichtet werden. Diese Integration wird voraussichtlich im Frühjahr 2012 durch einen Master Studenten der BFH E-Voting-Gruppe vorgenommen.

Threshold Decryption: Die Möglichkeit, verteilte Berechnungen mit nur einem Teil aller Parteien durchzuführen, kann nachträglich implementiert werden. Diesbezüglich ist zu prüfen, ob die bestehende Threshold Funktionalität von Verificatum genutzt werden soll, was jedoch tiefgreifende Änderungen an der Applikation zufolge hätte (siehe Anhang D).

Authentizifierung der Parteien: Die Applikation prüft nicht, ob es sich um die korrekte Partei handelt, also z.B. ob die richtige Person einen Mixing-Durchlauf durchführt. Um die Identität der verschiedenen Parteien zu verifizieren, könnten beispielsweise Zertifikate eingesetzt werden. Für die Stimmabgabe ist diese Methode allerdings nicht geeignet, da sie keine Coercion Resistance mehr bietet.

8.4 persönliche Erfahrungen

David Berger: Für mich bestand das Projekt aus mehreren Herausforderungen. Zum Einen war ich mich nicht gewohnt, wissenschaftliche Dokumentationen zu erarbeiten, deren Anspruch punkto Korrektheit von Wortlaut und Formeln wesentlich höher ist als bei den Arbeiten, welche ich während des restlichen Studiums verfasste. Zum Anderen war ich beruflich bisher stets in den Bereichen System und Network Engineering tätig. Mein Fachwissen in Programmierung beschränkte sich daher auf das, was ich während der Lehre und an der Fachhochschule gelernt hatte. Die Betrachtung des Themas E-Voting war für mich eine interessante und fordernde Erfahrung. Angesichts der Menge an Programmcode und der Grösse dieser Dokumentation schätze ich den Umfang unserer Arbeit als beträchtlich ein für ein 2-Personen Projekt. Ich hoffe, dass unsere Erkenntnisse für weitere Projekte der BFH E-Voting-Gruppe verwendet werden können.

Rolf Linder: Rückblickend bin ich stolz auf die Menge an Applikationscode, welcher innerhalb dieser Arbeit (und trotz dem längeren unbekannten Weg von Verificatum) entstanden ist, insbesondere in Anbetracht des anfänglichen Wissensstandes bzw. der Erfahrung. Die Grundkonzepte der angewendeten Kryptographie wurden zwar im vorangehenden Semester behandelt, es fühlte sich jedoch zeitweise ein wenig an, wie wenn man nach dem Erlernen des Fahrradfahrens nach einer halbjährigen Fahrpraxis an der Tour de France teilnehmen würde. Ich habe das Gefühl, das lehrreichste Semester abzuschliessen, obwohl die geringste Anzahl an Modulen belegt wurde. Die Umsetzung der theoretisch erlernten Kenntnisse der Programmierung war sehr lehrreich.

Literaturverzeichnis

[1] Dario Catalano Ari Jules and Markus Jacobsson. Coercion-Resistant Electronic Elections. 2005. [Referenziert in 1.1, 2.1]

[2] D. Chaum and T. P. Pedersen. Wallet Databases with Observers. In E. F. Brickell, editor, *CRYPTO'92, 12th Annual International Cryptology Conference on Advances in Cryptology*, LNCS 740, pages 89–105, Santa Barbara, USA, 1992. [Referenziert in 3.2]

[3] Wikipedia community (de.wikipedia.org). Babystep-Giantstep-Algorithmus. `http://de.wikipedia.org/wiki/Babystep-Giantstep-Algorithmus`. [Referenziert in 3]

[4] Wikipedia community (de.wikipedia.org). Diskreter Logarithmus. `http://de.wikipedia.org/wiki/Diskreter_Logarithmus`. [Referenziert in 3]

[5] Taher ElGamal. *A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms.* 1985. [Referenziert in 3.1]

[6] Reto Koenig Michael Schläpfer, Rolf Haenni and Oliver Spycher. Efficient Vote Authorization in Coercion-Resistant Internet Voting. 2011. [Referenziert in 1.1, 2.2]

[7] C. P. Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3):161–174, 1991. [Referenziert in 3.2]

[8] Anja Schütz. Die fünfte Schweiz auf dem steinigen Weg zum E-Voting. *Netzwoche 17/2011*, 2011. [Referenziert in 2]

[9] Warren D. Smith. New cryptographic voting scheme with best-known theoretical properties. In FEE'05, Workshop on Frontiers in Electronic Elections, Milan (Italy), 2005. [Referenziert in 2.2]

[10] swissinfo.ch. Wahlen 2011: Freud und Leid der Fünften Schweiz. `http://www.swissinfo.ch/ger/Politik/Wahlen_2011/5._Schweiz/Wahlen_2011:_Freud_und_Leid_der_Fuenften_Schweiz.html?cid=31430442`, 2011. [Referenziert in 2]

[11] G. Weber, R. Araujo, and J. Buchmann. On coercion-resistant electronic elections with linear work. In *Reliability and Security*, pages 908–916, In ARES'07, 2nd International Conference on Availability, Vienna (Austria), 2007. [Referenziert in 2.2]

[12] S. Weber. *Coercion-Resistant Cryptographic Voting: Implementing Free and Secret Electronic Elections.* VDM Verlag Saarbrücken (Germany), 2008. [Referenziert in 2.2]

[13] Douglas Wikstroem. Verificatum Documentation. `http://www.verificatum.com/verificatum/index.html`, 2011. [Referenziert in D]

Anhänge

Anhang A

Use Cases

Die folgenden Use Cases wurden im Rahmen der Projektinitialiserungsphase erstellt und dienten als Ausgangslage für die Implementierung. Die effektive Umsetzung innerhalb der Applikation kann vom hier beschriebenen Use Case abweichen. Als Vorgabe gilt in jedem Fall die Protokoll Spezifikation von SHKS11.

A.1 Stimmabgabe

Dieser Use Case reflektiert die Stimmabgabe eines Wählers (Voter V_i) im System.

Primärer Akteur: Wähler

Vorbedingungen:

- Wähler besitzt ein "Credential".
- Die öffentlichen Domänen-Parameter p, q und g sowie der öffentliche Schlüssel PK_T der Tallying Authorities sind bekannt.
- Die Voter Roll inkl. der Zuteilung der Voter IDs $(V_0, \ldots, V_i, \ldots, V_n)$ ist für den Wähler verfügbar.
- Eine Abstimmung ist aufgesetzt und freigegeben.

Ergebnisse und Nachbedingungen: Wähler hat Stimme abgegeben; Stimme wird entsprechend während der Phase "Vote Authorization" verarbeitet.

Ablauf:

1. Wähler bestimmt den zu wählenden Eintrag c_j aus der "Candidate Slate" C.
2. Wähler wählt zufällig zwei Werte r_1 und r_2 aus der Menge $\mathbb{Z}_q$ $(r_1, r_2 \in_R \mathbb{Z}_q)$
3. Wähler generiert das ElGamal Tupel A in der Form $A := (g^{r_1}, {PK_T}^{r_1} \cdot \sigma)$, wobei σ dem Credential vom Voter V (dem primären Akteur) entspricht
4. Wähler generiert das ElGamal Tupel B in der Form $B := (g^{r_2}, {PK_T}^{r_2} \cdot c_j)$, wobei c_j der Wahl des Wählers entspricht
5. Wähler generiert den NIZKP π_A als Beweis für die Kenntnis seines Credentials (Proof of Knowledge of Randomness r_1)
6. Wähler generiert den NIZKP π_B als Beweis für die Kenntnis seiner abgegebenen Stimme (Proof of Knowledge of Randomness r_2)
7. Wähler generiert den NIZKP π_C als Beweis für die Abgabe einer gültigen Stimme $(c_j \in C)$
8. Wähler wählt zufällig $\beta - 1$ Werte aus der Liste der Voter IDs und bildet die Menge I
9. Wähler fügt der Menge I die eigene Voter ID an (der Eintrag muss zufällig eingefügt werden, darf nicht stets dieselbe Position in der Menge haben)
10. Wähler übermittelt den Inhalt A, B, I, Π dem Bulletin Board $(\Pi = \pi_a + \pi_b + \pi_c)$

Häufigkeit: In der Regel ein Mal pro Wähler und Abstimmung.

A.2 Stimmen auszählen

Primärer Akteur: Tallying Authorities

Vorbedingungen:

- es haben sich 1 bis n Wähler registriert
- es wurden 1 bis n Stimmen abgegeben

Ergebnisse und Nachbedingungen: Sämtliche abgegebenen Stimmen wurden überprüft und gezählt, sofern gültig. Das Wahlergebnis liegt vor.

Ablauf:

1. Eliminieren der ungültigen Stimmen, indem die Beweise π_1, π_2 und π_3 überprüft werden. Stimmt einer oder mehrere der Beweise nicht, so wird die Stimme verworfen.

2. Eliminieren der doppelten Stimmen, damit kein Wähler absichtlich oder auch unabsichtlich mehrere Male stimmen kann. Zu diesem Zweck wird das Verfahren von Smith & Weber eingesetzt. Dieses sieht vor, dass verschlüsselte ElGamal Tupel, welches das Credential σ des Wählers enthält, mit einem den Tallying Authorities bekannten Zufallswert z verblindet wird:

$$E_1{}^z = E_T(\sigma_1{}^z, r_1{}^z)$$

$$E_2{}^z = E_T(\sigma_2{}^z, r_2{}^z)$$

$$\dots$$

$$E_n{}^z = E_T(\sigma_n{}^z, r_n{}^z)$$

   Die resultierenden, sogenannten Blindwerte $\sigma_n{}^z$ werden anschliessend in einer Hash-Tabelle abgespeichert und miteinander verglichen. Stimmt $\sigma_i{}^z$ mit $\sigma_j{}^z$ überein, so hat der Wähler doppelt abgestimmt und eine der beiden Stimmen wird markiert.

3. Replikation der abgegebenen, noch gültigen Stimmen. In diesem Schritt wird jede Stimme, bestehend aus den Tupeln (A, B, S_n) in β Stimmen repliziert. Jede Stimme führt folglich $\beta - 1$ (absichtlich) gefälschten und einer gültigen Stimme. β stellt dabei die Grösse des sogenannten Anonymitätssets dar, welches als Sicherheits-Parameter den Grad der in der Einleitung definierten Coercion Resistance definiert.

4. Die Liste mit allen replizierten Stimmen wird an ein Verifiable Reencryption Mix-Network weiter gegeben, welches die Tupel $(A', B', S_n{}')$ sowie die entsprechenden Beweise dazu liefert.

5. Elimineren der gefälschten Stimmen. Die Tallying Authorities führen nun für jedes Tupel (A', S_n) den Plaintext Equivalence Test (PET) durch. Stimmen, bei denen der Algorithmus "false" zurückgibt, werden entsprechend eliminiert.

6. Die verbleibenden Stimmen werden als gültig angesehen. Das ElGamal Tupel B' wird von den Tallying Authorities gemeinsam entschlüsselt und die Stimme entsprechend gezählt.

Häufigkeit: Am Ende jeder Abstimmung.

A.3 Abstimmung einrichten

Primärer Akteur: Administrator

Vorbedingungen: Die von den Wahlbehörden (Registrar Authority) definierte Wählerliste (die Voter Roll) ist definiert.

Ergebnisse und Nachbedingungen: Es ist eine Abstimmung eingerichtet und bereit zur Aufnahme von Wählerstimmen. Das Bulletin Board befindet sich im Status "Vote Casting".

Ablauf:

1. Der Administrator setzt den Titel der Abstimmung.
2. Der Administrator definiert die Wahloptionen (Candidate Slate).
3. Die Tallying Authorities berechnen gemeinsam den öffentlichen Schlüssel für das Bulletin Board.
4. Die Wahlbehörden publizieren die Voter Roll auf dem Bulletin Board.
5. Der Administrator setzt den Status vom Bulletin Board auf "Vote Casting".

Häufigkeit: Am Anfang jeder Abstimmung.

A.4 Registrierung eines Wählers

Primärer Akteur: Wähler

Vorbedingungen: Der Wähler muss die gesetzlichen Bestimmungen für das Wahlrecht erfüllen.

Ergebnisse und Nachbedingungen: Der Wähler hat sich erfolgreich registriert und kann fortan an Abstimmungen teilnehmen.

Ablauf: Die Registrierung des Wählers muss gemäss dem Protokoll über einen sogenannten "Untappable Channel" geschehen. Eine verschlüsselte Verbindung reicht dafür nicht aus, stattdessen wird eine physikalische, abhörsichere Punkt-zu-Punkt Verbindung zwischen Wähler und Wahlbehörde (Registrar Authority) vorausgesetzt, was in der Praxis nicht realistisch ist. Wir nehmen daher an, dass der Wähler stattdessen persönlich bei der Wahlbehörde vorstellig wird (analog dem heutigen Verfahren für die Beantragung eines Passes oder einer Identitätskarte) und dort seinen digitalen Stimmrechtsausweis erhält.

Häufigkeit: Einmalig pro Wähler (vor der ersten Stimmabgabe).

A.5 Abstimmungsresultat prüfen

Primärer Akteur: Verifier

Vorbedingungen: Um das Abstimmungsresultat überprüfen zu können, müssen vorher die Tallying Authorites eine Stimmenauszählung vorgenommen haben, damit ein Vergleich der Resultate möglich ist.

Ergebnisse und Nachbedingungen: Der Verifier hat das Abstimmungsresultat erfolgreich verifiziert. Sollte er nicht zum selben Resultat kommen, liegt ein Fehler bei der Auszählung der Tallying Authority vor.

Ablauf:

- überprüfen des gemeinsam errechneten, öffentlichen Schlüssels der Tallying Authorities
- überprüfen der Beweise für die Teilschlüssel der Tallying Authorities
- überprüfen der Beweise für die Teilverblindungswerte der Tallying Authorities
- überprüfen der Beweise für die Teilverblindungen von ElGamal Tupeln

- überprüfen der Phase “Invalid Vote Elimination”
- überprüfen der Phase “Duplicate Vote Elimination”
- überprüfen der Phase “Ballot Replication”
- rudimentäre Prüfung des Reencryption Mix-Networks
- überprüfen der Phase “Fake Vote Elimination”
- überprüfen der Phase “Tallying”

Häufigkeit: Auf Anfrage des Überprüfers.

Anhang B

Proof of Validity

Es soll bewiesen werden, dass eine Verschlüsselung einen Wert aus einer Liste von n gültigen Werten darstellt, wobei n beliebig gross sein kann.

Eine Verschlüsselung c besteht aus einem ElGamal Tupel der Form $(a, b) = (g^r, m \cdot y^r)$, wobei $m \in \{m_1, \ldots, m_n\}$ gilt. Die Liste der möglichen Verschlüsselungen hat die Form

$$(g^r, \frac{m}{m_1} \cdot y^r), \ldots, (g^r, \frac{m}{m_n} \cdot y^r)$$

resp.

$$(a, b_1), \ldots, (a, b_n)$$

wobei

$$b_n = \frac{b}{m_n}$$

gilt.

Das Problem kann daher auf den mehrfachen Beweis "Proof of Knowledge of Equality of Discrete Logarithm" reduziert werden:

$$(\log_g a = \log_y b_1) \text{ OR } \ldots \text{OR } (\log_g a = \log_y b_n)$$

Da der Prover diesen Beweis nur für ein einziges $m \in \{m_1, \ldots, m_n\}$ erbringen kann, muss er alle anderen Beweise "simulieren".

Angenommen der Prover hat die Nachricht m_i verschlüsselt und will beweisen, dass $m_i \in \{m_1, \ldots, m_n\}$ gilt. Der Beweis P_i wird analog dem im Paragraph "Proof of Knowledge of Equality of Discrete Logarithm" beschriebenen Verfahren berechnet (Beweis vom Element m_i). Das Commitment der anderen Beweise ($P_1, \ldots, P_n$, für $n \neq i$), wird hingegen nicht anhand von ω berechnet, sondern durch die zuerst zufällig gewählten Werte c_n und s_n berechnet. Folgend das Vorgehen im Detail.

Der Prover wählt zufällig den Wert $\omega_i \in_R \mathbb{Z}_q$ und berechnet damit die beiden Commitments

$$t_{\alpha i} = g^{\omega_i}, t_{\beta i} = h^{\omega_i}$$

für den Beweis P_i. Zur Simulation der anderen Beweise $P_1, \ldots, P_n$ wählt er jeweils zufällig $c_n, s_n \in_R \mathbb{Z}_q$. Die beiden Commitments errechnet er danach wie folgt:

$$t_{\alpha n} = g^{s_n} \cdot {y_{\alpha n}}^{-c_n}, \quad t_{\beta n} = h^{s_n} \cdot {y_{\beta n}}^{-c_n}$$

Nun wird bei der nicht-interaktiven Variante wiederum ein Hashwert C generiert, welcher alle Commitments, sowie alle y-Werte enthählt, für welche der Logarithmus bewiesen werden soll ($n \neq i$):

$$C = H(t_{\alpha 1}, t_{\beta 1}, y_{\alpha 1}, y_{\beta 1}, \ldots, t_{\alpha n}, t_{\beta n}, y_{\alpha n}, y_{\beta n})$$

Der Challenge c_i berechnet sich aus der Differenz zwischen C und den simulierten Challenges $c_1, \ldots, c_n$ ($n \neq i$), also

$$c_i = C - \sum_{j=1, j \neq i}^{n} c_j$$

Der Prover berechnet schliesslich $s_1 = \omega_1 + c_1 \cdot x_1$ und publiziert die Liste aller Commitments, Challenges und der Responses der einzelnen Teilbeweise (inklusive der Elemente von i):

$$(t_{\alpha 1}, t_{\beta 1}, c_1, s_1, \ldots, t_{\alpha n}, t_{\beta n}, c_n, s_n)$$

Die Überprüfung des Beweises durch einen Verifier erfolgt in mehreren Schritten:

1. überprüfen jedes einzelnen Beweises $P_1, \ldots, P_n$:

$$t_{\alpha n} \cdot {y_{\alpha n}}^{c_n} \stackrel{?}{=} g^{s_n} \tag{B.1}$$

$$t_{\beta n} \cdot {y_{\beta n}}^{c_n} \stackrel{?}{=} h^{s_n} \tag{B.2}$$

$$H(t_{\alpha n}, t_{\beta n}, y_{\alpha n}, y_{\beta n}) \stackrel{?}{=} c_n \tag{B.3}$$

2. überprüfen, ob die Summe der Challenges $c_1, \ldots, c_n$ dem Hashwert der Commitments und y-Werte entsprechen:

$$H(t_{\alpha 1}, t_{\beta 1}, y_{\alpha 1}, y_{\beta 1}, \ldots, t_{\alpha n}, t_{\beta n}, y_{\alpha n}, y_{\beta n}) \stackrel{?}{=} c_1 + \ldots + c_n \tag{B.4}$$

Damit überprüft ein Verifier einerseits, dass der Prover einen Beweis erbringen musste (nicht alle Beweise simulieren konnte) und dass alle Beweise korrekt sind.

Anhang C

java.util.Collections.shuffle()

Der nachfolgend dargestellte Algorithmus ist im Java Runtime Environment enthalten und wird durch Vertauschung der Elemente bei der Bildung vom Anonymity Set benutzt.

```
public static void shuffle(List<?> list, Random rnd) {
    int size = list.size();
    if (size < SHUFFLE_THRESHOLD || list instanceof RandomAccess) {
        for (int i=size; i>1; i--)
            swap(list, i-1, rnd.nextInt(i));
    } else {
        Object arr[] = list.toArray();

        // Shuffle array
        for (int i=size; i>1; i--)
            swap(arr, i-1, rnd.nextInt(i));

        // Dump array back into list
        ListIterator it = list.listIterator();
        for (int i=0; i<arr.length; i++) {
            it.next();
            it.set(arr[i]);
        }
    }
}
```

Anhang D

Verificatum

Bei Verificatum [13] handelt es sich, wie im Abschnitt 1.1.4 bereits erwähnt, um eine Verifiable Reencryption Mix-Network Implementation in Java. Verificatum ist ein Forschungsprojekt an der School of Computer Science and Communication des Royal Institute of Technology in Schweden.

Ein erster Test der aktuellen, frei verfügbaren Version von Verificatum schlug fehl. Die Absolventen entschieden sich daher im Oktober dafür, mit dem Hauptautor von Verificatum, Prof. Dr. Douglas Wikstroem, Kontakt aufzunehmen. Professor Wikstroem zeigte sich sehr hilfsbereit und richtete einen Testbenutzer auf einem Server an der Universität in Schweden ein, mit welchem Verificatum und die dazu gehörende Demo schliesslich getestet werden konnten.

In der Folge zeigte sich relativ rasch, dass Verificatum einen enormen Funktionsumfang besitzt, welcher weit über die geforderte Funktionalität in dieser Arbeit hinaus geht. So beinhaltet Verificatum neben der Funktion eines Verifiable Reencryption Mix-Network unter anderem auch eine verteilte Schlüsselgenerierung, eine ebenfalls verteilte Entschlüsselung des Inputs und eine Verifier Komponente, welche es ermöglicht, sämtliche erbrachten Beweise zu überprüfen. Anzumerken ist ausserdem, dass alle verteilten Operationen im Threshold Verfahren ausgeführt werden, wobei die minimale Anzahl der gültigen Teilnehmer zusammen mit vielen anderen Parametern in entsprechenden Konfigurationsdateien gespeichert werden.

Wie im 1. Punkt der Anforderungen an das Reencryption Mix-Net erwähnt, liefert die dem Mixing vorhergehende Phase "Ballot Replication" jeweils 3 ElGamal Tupel pro Stimme als Input für das Mix-Network. Professor Wikstroem bestätigte auf Anfrage, dass Verificatum standardmässig nur ein Tupel als Input erwartet. Er erklärte sich aber freundlicherweise dazu bereit, Verificatum geringfügig anzupassen, um ein Input von 3 Tupel zu erlauben, wobei die Anzahl der Tupel über die Konfigurationsdatei gesetzt werden kann. Nach einem ausführlichen Gespräch im Oktober wurden folgende Probleme bezüglich der Verwendung von Verificatum in dieser Arbeit ermittelt:

- Verificatum generiert selbständig ein entsprechendes Schlüsselpaar für die Ver- und Entschlüsselung der Tupel. Dies ist aus Sicht unseres Projekts problematisch, da für die Reencryption der Stimmen der von den Tallying Authorities generierte Schlüssel verwendet werden muss, ansonsten kann in der Phase "Tallying" keine Entschlüsselung mehr vorgenommen werden.
- Verificatum verwendet sowohl für Input als auch für Output entsprechende Dateien im Textformat, wobei die Tupel in einem Byte Tree Format abgespeichert werden. Unsere Applikation beruht hingegen auf der Persistierung von Java Objekten in Verbindung mit einer Datenbank (JPA). Für den Datenaustausch mit Verificatum müsste daher entweder Verificatum entsprechend erweitert werden, oder es müsste eine Schnittstelle für das Ein- und Auslesen der Dateien von Verificatum entwickelt werden.
- Verificatum liefert, wie bereits erwähnt, eine eigene Verifier Komponente zur Überprüfung der Beweise mit. Die Möglichkeit, die Beweise durch eine externe Applikation überprüfen zu lassen ist gemäss Professor Wikstroem in Planung, aber noch nicht fertiggestellt. Für unser Projekt hiesse das, dass der gesamte Bereich des Reencryption Mix-Nets inklusive der generierten Beweise eine Art "Blackbox" darstellt, deren korrekte Funktion nicht objektiv beurteilt werden kann.

Nachdem Professor Wikstroem kurz ins Protokoll von SHKS11 eingeführt wurde, lautete sein Vorschlag, dass wir Verificatum dahingehend weiterentwickeln, dass insbesondere die verteilten Aktivitäten wie z.B. der PET, als zusätzliche Komponente von Verificatum ausgeführt werden könnten. Er unterstrich mehrere Male, dass er daran interessiert sei, dass Verificatum genutzt und weiter entwickelt werde und er versprach auch, die Absolventen bei den entsprechenden Anpassungen zu unterstützen.

Das Thema Mix-Network wurde einige Wochen vertagt und schliesslich im November wieder aufgegriffen, da der Bereich Vote Authorization bereits relativ weit fortgeschritten war und eine Entscheidung getroffen

werden musste. Prof. Wikstroem hatte in der Zwischenzeit unabhängig von unserem Projekt Verificatum dahingehend erweitert, dass die Funktionalität der einzelnen Mixer auf "nur-mixen" oder "nur-entschlüsseln" eingeschränkt werden konnte. Desweiteren veröffentlichte er ein Benutzerhandbuch für Verificatum, welches unter anderem auch die Integration von Verificatum mittels eines entsprechenden Interfaces beschrieb. Die Studenten haben dieses Benutzerhandbuch studiert und Prof. Wikstroem die von Ihnen verwendeten Datenstrukturen "Field", "FieldElement", "ElGamalTuple" und "ElGamalSystem" (siehe Abschnitt 4.12.1) erklärt. Ende November wurde die Verwendung von Verificatum erneut im Rahmen einer Skype Session mit Prof. Wikstroem besprochen und die erforderlichen Schritte definiert:

1. implementieren einer Import- und Export-Komponente, die es erlaubt, die von uns generierten ElGamal Tupel in einer Datei zu speichern und (nach dem Mixing) wieder zu lesen
2. implementieren des Interfaces "MixNetElGamalInterface", resp. der folgenden Methoden:
   - writePublicKey - Öffentlicher Schlüssel in eine Datei schreiben.
   - readPublicKey - Öffentlicher Schlüssel aus einer Datei lesen inkl. der zugehörigen Gruppe.
   - writeCiphertexts - Verschlüsselte Stimmen in eine Datei schreiben.
   - readCiphertexts - Verschlüsselte Stimmen aus einer Datei lesen.
   - decodePlaintexts - Entschlüsseln der Stimmen und speichern in eine Datei.

Während der erste Schritt, resp. Export und Import über eine Datei, relativ einfach zu realisieren wäre (die entsprechenden Klassen implementieren bereits das Interface "Serializable"), ist der zweite Schritt aufgrund der unterschiedlichen Datenstruktur relativ aufwändig. Prof. Wikstroem bekräftige, dass er uns bei der Implementation der Interfaces helfen würde. Er schlug ausserdem vor, nicht nur die Schlüsselgenerierung und das Mixing an Verificatum zu delegieren, sondern auch die Threshold Decryption und den PET. Es zeichnete sich ab, dass bei einer Verwendung von Verificatum die beiden Rollen "Tallying Authorities" und "Mixer" verschmelzen, was nach Ansicht von Prof. Wikstroem richtig wäre, in den Papers JCJ05 und SHKS11 allerdings so nicht vorgesehen ist. Nach Ansicht der Projektbetreuer wäre dies nicht konform aber grundsätzlich vertretbar und daher kein Hindernis, Verificatum einzusetzen. Eine Entflechtung könne nach Abschluss der Arbeit vorgenommen werden.

Studenten und Betreuer waren sich einig, dass Verificatum ein äusserst interessantes Projekt ist, da es nach unserem Wissensstand die erste, frei verfügbare Reencryption Mix-Network Implementation in Java darstellt. Folgende Gründe führten jedoch letztendlich zum Entscheid, Verificatum nicht zu integrieren:

- Der Aufwand für den zu realisierenden Datenaustausch zwischen Verificatum und unserer Applikation und die damit verbundene Konvertierung der Datenstrukturen ist nicht abschätzbar. Eine solche Entwicklung noch in der 10. von 16 Wochen in Angriff zu nehmen wäre angesichts der ausstehenden Punkte ein erhebliches Projektrisiko.
- Die Abhängigkeit von einer Fremdkomponente generiert einen neuen Unsicherheitsfaktor und führt zudem dazu, dass nicht objektiv überprüft werden kann, ob das Mix-Network wurde bereits genannt. Es wird ausserdem vermutet, dass sich insbesondere die Datenkonvertierung zwischen Verificatum und unserer Applikation negativ auf die Performance der Vote Authorization auswirken könnte.
- Verificatum konnte bisher lediglich auf dem erwähnten Red Hat Server von Prof. Wikstroem getestet werden. Eine lokale Installation schlug fehl. Um Verificatum im Rahmen einer Demonstration verlässlich nutzen zu können, bedürfte es daher auch an weiteren Arbeiten im Bereich eines Linux Systems.
- Um Verificatum sinnvoll einzusetzen, hätte unsere Applikation sinnvollerweise in Verificatum integriert werden sollen anstatt umgekehrt. Eine solche Änderung des Konzepts würde aber den Projektauftrag als solcher in Frage stellen und käme zum jetzigen Zeitpunkt ohnehin nicht mehr in Frage.

Aus diesen Gründen wurde mit den Projektbetreuern vereinbart, eine eigene Mix-Network Implementation einzusetzen. Da die Studenten nicht über die nötigen Beweistechniken verfügen um die Funktionsweise des Mix-Networks zu beweisen, wird es sich um eine "nicht-überprüfbare" Mix-Network Implementation handeln, welche folglich keine Beweise erbringt. Die Projektbetreuer haben bestätigt, dass diese Funktionseinschränkung keine negativen Auswirkungen auf die Bewertung des Projekts haben wird und die Bemühungen entsprechend honoriert werden. Es wurde mit Prof. Wikstroem vereinbart, dass die Integration von Verificatum voraussichtlich im Frühjahr 2012 ausserhalb dieser Bachelor Thesis angegangen wird.

Anhang E

SQL Dump

E.1 Datenbank “BulletinBoard”

```
-- phpMyAdmin SQL Dump
-- version 2.11.9.2
-- http://www.phpmyadmin.net
--
-- Host: localhost
-- Erstellungszeit: 05. Januar 2012 um 16:55
-- Server Version: 5.0.67
-- PHP-Version: 5.2.6

SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";

--
-- Datenbank: `BulletinBoard`
--

-- --------------------------------------------------------

--
-- Tabellenstruktur fuer Tabelle `BALLOTENTRY`
--

CREATE TABLE IF NOT EXISTS `BALLOTENTRY` (
  `ID` bigint(20) NOT NULL auto_increment,
  `A` longblob,
  `B` longblob,
  `I` longblob,
  `PI1` longblob,
  `PI2` longblob,
  `PI3` longblob,
  `S` longblob,
  `STATE` int(11) default NULL,
  `TIMESTAMP` datetime default NULL,
  PRIMARY KEY  (`ID`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=1282 ;

-- --------------------------------------------------------

--
-- Tabellenstruktur fuer Tabelle `BLINDEDRESULTENTRY`
--

CREATE TABLE IF NOT EXISTS `BLINDEDRESULTENTRY` (
  `ID` bigint(20) NOT NULL auto_increment,
  `BALLOTID` bigint(20) default NULL,
  `BLINDEDRESULT` longblob,
```

```
  `INSTANCEID` int(11) default NULL,
  `INSTANCETYPE` int(11) default NULL,
  `STATE` int(11) default NULL,
  `TIMESTAMP` datetime default NULL,
  PRIMARY KEY  (`ID`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=232 ;

-- --------------------------------------------------------

--
-- Tabellenstruktur fuer Tabelle `BLINDINGCOMMITMENTENTRY`
--

CREATE TABLE IF NOT EXISTS `BLINDINGCOMMITMENTENTRY` (
  `ID` bigint(20) NOT NULL auto_increment,
  `INSTANCEID` int(11) default NULL,
  `INSTANCETYPE` int(11) default NULL,
  `KNOWNZ` longblob,
  `STATE` int(11) default NULL,
  `TIMESTAMP` datetime default NULL,
  `ZCOMMITMENT` longblob,
  PRIMARY KEY  (`ID`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=4 ;

-- --------------------------------------------------------

--
-- Tabellenstruktur fuer Tabelle `BLINDINGTUPLEENTRY`
--

CREATE TABLE IF NOT EXISTS `BLINDINGTUPLEENTRY` (
  `ID` bigint(20) NOT NULL auto_increment,
  `BALLOTID` bigint(20) default NULL,
  `BLINDINGTUPLE` longblob,
  `INSTANCEID` int(11) default NULL,
  `INSTANCETYPE` int(11) default NULL,
  `PROOF` longblob,
  `STATE` int(11) default NULL,
  `TIMESTAMP` datetime default NULL,
  PRIMARY KEY  (`ID`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=694 ;

-- --------------------------------------------------------

--
-- Tabellenstruktur fuer Tabelle `CANDIDATESLATEENTRY`
--

CREATE TABLE IF NOT EXISTS `CANDIDATESLATEENTRY` (
  `ID` bigint(20) NOT NULL auto_increment,
  `DESCRIPTION` varchar(255) default NULL,
  `STATE` int(11) default NULL,
  `TIMESTAMP` datetime default NULL,
  PRIMARY KEY  (`ID`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=3 ;

-- --------------------------------------------------------
```

```
--
-- Tabellenstruktur fuer Tabelle `DATAENTRY`
--

CREATE TABLE IF NOT EXISTS `DATAENTRY` (
  `ID` bigint(20) NOT NULL auto_increment,
  `IDENTITY` varchar(255) default NULL,
  `SIGMA` longblob,
  `TIMESTAMP` datetime default NULL,
  PRIMARY KEY  (`ID`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=21 ;

-- --------------------------------------------------------

--
-- Tabellenstruktur fuer Tabelle `DECRYPTINGTUPLEENTRY`
--

CREATE TABLE IF NOT EXISTS `DECRYPTINGTUPLEENTRY` (
  `ID` bigint(20) NOT NULL auto_increment,
  `BALLOTID` bigint(20) default NULL,
  `DECRYPTINGTUPLE` longblob,
  `INSTANCEID` int(11) default NULL,
  `INSTANCETYPE` int(11) default NULL,
  `PROOF` longblob,
  `STATE` int(11) default NULL,
  `TIMESTAMP` datetime default NULL,
  PRIMARY KEY  (`ID`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=745 ;

-- --------------------------------------------------------

--
-- Tabellenstruktur fuer Tabelle `DUPLICATEVOTEENTRY`
--

CREATE TABLE IF NOT EXISTS `DUPLICATEVOTEENTRY` (
  `ID` bigint(20) NOT NULL auto_increment,
  `STATE` int(11) default NULL,
  `TIMESTAMP` datetime default NULL,
  `MARKEDENTRY_ID` bigint(20) default NULL,
  PRIMARY KEY  (`ID`),
  KEY `FK_DUPLICATEVOTEENTRY_MARKEDENTRY_ID` (`MARKEDENTRY_ID`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=2 ;

-- --------------------------------------------------------

--
-- Tabellenstruktur fuer Tabelle `FAKEVOTEENTRY`
--

CREATE TABLE IF NOT EXISTS `FAKEVOTEENTRY` (
  `ID` bigint(20) NOT NULL auto_increment,
  `STATE` int(11) default NULL,
  `TIMESTAMP` datetime default NULL,
  `MARKEDENTRY_ID` bigint(20) default NULL,
```

```
  PRIMARY KEY  (`ID`),
  KEY `FK_FAKEVOTEENTRY_MARKEDENTRY_ID` (`MARKEDENTRY_ID`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=194 ;

-- --------------------------------------------------------

--
-- Tabellenstruktur fuer Tabelle `INVALIDVOTEENTRY`
--

CREATE TABLE IF NOT EXISTS `INVALIDVOTEENTRY` (
  `ID` bigint(20) NOT NULL auto_increment,
  `STATE` int(11) default NULL,
  `TIMESTAMP` datetime default NULL,
  `MARKEDENTRY_ID` bigint(20) default NULL,
  PRIMARY KEY  (`ID`),
  KEY `FK_INVALIDVOTEENTRY_MARKEDENTRY_ID` (`MARKEDENTRY_ID`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;

-- --------------------------------------------------------

--
-- Tabellenstruktur fuer Tabelle `MARKERENTRY`
--

CREATE TABLE IF NOT EXISTS `MARKERENTRY` (
  `MARKEDENTRY_ID` bigint(20) default NULL,
  KEY `FK_MARKERENTRY_MARKEDENTRY_ID` (`MARKEDENTRY_ID`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- Tabellenstruktur fuer Tabelle `MIXEDBALLOTENTRY`
--

CREATE TABLE IF NOT EXISTS `MIXEDBALLOTENTRY` (
  `ID` bigint(20) NOT NULL auto_increment,
  `INSTANCEID` int(11) default NULL,
  `INSTANCETYPE` int(11) default NULL,
  `STATE` int(11) default NULL,
  `TIMESTAMP` datetime default NULL,
  `MIXEDBALLOT_ID` bigint(20) default NULL,
  PRIMARY KEY  (`ID`),
  KEY `FK_MIXEDBALLOTENTRY_MIXEDBALLOT_ID` (`MIXEDBALLOT_ID`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=1051 ;

-- --------------------------------------------------------

--
-- Tabellenstruktur fuer Tabelle `PUBKEYENTRY`
--

CREATE TABLE IF NOT EXISTS `PUBKEYENTRY` (
  `ID` bigint(20) NOT NULL auto_increment,
  `INSTANCEID` int(11) default NULL,
  `INSTANCETYPE` int(11) default NULL,
```

```
  `KNOWNX` longblob,
  `PUBKEY` longblob,
  `STATE` int(11) default NULL,
  `TIMESTAMP` datetime default NULL,
  PRIMARY KEY  (`ID`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=4 ;

-- --------------------------------------------------------

--
-- Tabellenstruktur fuer Tabelle `TALLYINGRESULTENTRY`
--

CREATE TABLE IF NOT EXISTS `TALLYINGRESULTENTRY` (
  `ID` bigint(20) NOT NULL auto_increment,
  `COUNT` int(11) default NULL,
  `STATE` int(11) default NULL,
  `TIMESTAMP` datetime default NULL,
  `MARKEDENTRY_ID` bigint(20) default NULL,
  PRIMARY KEY  (`ID`),
  KEY `FK_TALLYINGRESULTENTRY_MARKEDENTRY_ID` (`MARKEDENTRY_ID`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=3 ;

-- --------------------------------------------------------

--
-- Tabellenstruktur fuer Tabelle `VOTERROLLENTRY`
--

CREATE TABLE IF NOT EXISTS `VOTERROLLENTRY` (
  `ID` bigint(20) NOT NULL auto_increment,
  `IDENTITY` varchar(255) default NULL,
  `S` longblob,
  `STATE` int(11) default NULL,
  `TIMESTAMP` datetime default NULL,
  `VOTERID` int(11) default NULL,
  PRIMARY KEY  (`ID`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=18 ;
```

E.2 Datenbank “Registrar”

```
-- phpMyAdmin SQL Dump
-- version 2.11.9.2
-- http://www.phpmyadmin.net
--
-- Host: localhost
-- Erstellungszeit: 05. Januar 2012 um 17:20
-- Server Version: 5.0.67
-- PHP-Version: 5.2.6

SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";

--
-- Datenbank: `Registrar`
--

-- --------------------------------------------------------
```

```
--
-- Tabellenstruktur fuer Tabelle ‘DATAENTRY‘
--

CREATE TABLE IF NOT EXISTS ‘DATAENTRY‘ (
  ‘ID‘ bigint(20) NOT NULL auto_increment,
  ‘IDENTITY‘ varchar(255) default NULL,
  ‘SIGMA‘ longblob,
  ‘TIMESTAMP‘ datetime default NULL,
  PRIMARY KEY  (‘ID‘)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=21 ;
```

Anhang F

Verwendete PC Systeme zur Simulation

Für die Simulationsdurchläufe wurden die nachfolgend genauer spezifizierten Systeme eingesetzt:

Desktop PC: 3GHz Intel Core 2 Duo CPU, 4GB RAM, Linux Betriebssystem (64Bit)

Laptop PC: 2.27GHz Intel Core i5 CPU, 4GB RAM, Linux Betriebssystem (64Bit)

MacBook: 2.16GHz Intel Core 2 Duo CPU, 2GB RAM, Mac OS X Betriebssystem (32Bit)