Berner Fachhochschule
Technik und Informatik

Pascal Witschi & Stefan Johner

Bachelor Thesis in Information Technology

Sommer Semester 2009

Wählen und Zählen: Realisierung zweier Kernkomponenten des TrustVote-Protokolls für elektronische Abstimmungen

Betreuer:

- Prof. Dr. Rolf Haenni (rolf.haenni@bfh.ch)
  Berner Fachhochschule
- Prof. Dr. Stephan Fischli (stephan.fischli@bfh.ch)
  Berner Fachhochschule

Experte:

- Dr. Joachim Wolfgang Kaltz (joachimwolfgang.kaltz@postfinance.ch)
  Postfinance

Zusammenfassung

Das TrustVote-Protokoll ist ein von der Berner Fachhochschule entwickelts Protokoll für elektronische Abstimmungen und Wahlen. Realisiert wurden zwei Kernkomponenten dieses Protokolls, die Voter und die Tallier Applikation. Die Voter Applikation ermöglicht die bequeme und sichere Stimmabgabe auf einem Computer. Mit Hilfe der Tallier Applikation soll das Ergebnis der Wahl aufgrund der veröffentlichten elektronischen Wahlzettel für die Stimmenden wie auch die Wahlbehörde nachvollziehbar sowie überprüfbar sein.

Inhaltsverzeichnis

Kapitel 1

Einführung

1.1 Hintergrund

Elektronische Abstimmungen und Wahlen sind weltweit ein aktuelles Thema. Die Behörden erhoffen sich hauptsächlich steigende Wahlbeteiligungen sowie eine höhere Kosteneffizienz. Die meisten Projekte im Bereich der elektronischen Abstimmungen und Wahlen zielen darauf ab, den Stimmberechtigten die Stimmabgabe über das Internet zu ermöglichen. Diese Form der elektronischen Abstimmung und Wahl wird auch als I-Voting oder Remote E-Voting bezeichnet. Die Schweiz übernimmt in diesem Bereich eine Pionierrolle. In den Kantonen Genf, Neuenburg sowie Zürich existieren bereits seit einigen Jahren erfolgreiche I-Voting Pilotprojekte[1].

Elektronische Abstimmungen und Wahlen stellen ausserordentliche Anforderungen an die Sicherheit. In der Kryptographie wird seit langem nach Wahlprotokollen geforscht, welche möglichst vielen, sich zum Teil (scheinbar) widersprechenden Anforderungen genügen sollen. Die Herausforderungen sind die Wahrung des Wahlgeheimnisses bei gleichzeitiger Nachvollziehbarkeit und Unverfälschbarkeit der Wahl. Insbesondere sollen die Protokolle aber auch die Forderung nach Offenheit und Transparenz erfüllen. Zudem ist sicherzustellen, dass die Wähler[1] tatsächlich Vertrauen in die ergriffenen Sicherheitsmassnahmen haben. Dies kann angesichts der technischen Komplexität eine schwierige Aufgabe sein.

Ausserdem stellen sich bei elektronischen Abstimmungen und Wahlen auch verschiedene andere, nicht technische Herausforderungen. So könnte zum Beispiel durch die (mögliche) Vereinfachung des Wahlgangs durch I-Voting eine Entwertung des Wählens stattfinden. Stimmen könnten verstärkt unreflektiert abgegeben werden.

1.2 Ausgangslage

An der Berner Fachhochschule werden im Rahmen eines Forschungsprojektes verschiedene Arbeiten im Bereich von elektronischen Wahlen und Abstimmungen durchgeführt. Das Ziel besteht darin, ein E-Voting-System zu entwerfen, welches auf die

[1] Die aus schreibtechnischen Gründen gewählte männliche Form gilt selbstverständlich auch für die weibliche Form.

besonderen Voraussetzungen und Bedürfnisse der Schweiz zugeschnitten ist. Daraus wurde das sogenannte TrustVote-Protokoll[2] entwickelt, welches viele der gewünschten Sicherheitsanforderungen erfüllt.

Im Rahmen des Swiss E-Voting-Workshops 2009[2] wurde eine konkrete Implementierung dieses Systems zu Demonstrationszwecken angestrebt. Zwei Elemente des TrustVote-Systems sind für potentielle Wähler relevant. Eine Voter Applikation ermöglicht die bequeme und sichere Stimmabgabe auf einem Computer. Die von der Voter Applikation verschlüsselte und von den Authorities blind signierte Stimme wird danach auf dem Public Board publiziert. Mit Hilfe einer Tallier Applikation soll, nachdem die Teilschlüssel am Ende der Stimmabgabe von den Key Collectors veröffentlicht wurden, das Ergebnis der Wahl aufgrund der veröffentlichten Wahlzettel für die Stimmenden wie auch die Wahlbehörde nachvollziehbar sowie überprüfbar sein. Abbildung 1.1 gibt eine vereinfachte Übersicht über das TrustVote-Protokoll.

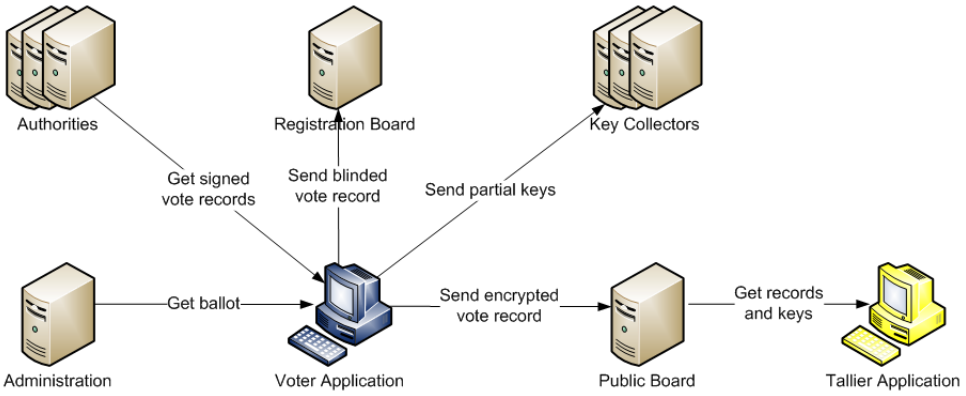

Abbildung 1.1: Übersicht TrustVote-Protokoll (vereinfacht)

Beide Applikationen erfordern die Kommunikation mit anderen Komponenten des Systems, wobei vorgegebene Protokolle und Schnittstellen zu berücksichtigen sind. Zudem kommen verschiedenste kryptographische Funktionen, wie unter anderem (t, N)-Threshold Blind Signature[3] und Shamir Secret Sharing[4], zum Einsatz.

Nebst den Voter und Tallier Applikationen wurden im Rahmen der Umsetzung des Protokolls für den Swiss E-Voting-Workshop 2009 auch alle weiteren Komponenten des TrustVote-Protokolls implementiert.

1.3 Leserkreis

Dieses Dokument richtet sich an den Experten, die Betreuer sowie an alle interessierten Personen, die sich mit diesem Thema auseinandersetzen wollen. Ausserdem soll es als Grundlage für die Weiterentwicklung der Anwendungen dienen.

[2]Swiss E-Voting Workshop 2009 vom 5. Juni 2009 in Münchenwiler bei Murten (http://www.e-voting-cc.ch/index.php/de/workshop09

1.4 Aufbau des Dokuments

Das vorliegende Dokument beschreibt das Design der Applikationen sowie deren Realisierung. Weiter soll erläutert werden, wie die Systeme getestet wurden und welche Ergebnisse die Tests lieferten. Im letzten Kapitel werden im Rahmen einer Besprechung die Zielerreichung sowie die Lessons Learned thematisiert.

1.5 Inhalt der CD-ROM

Im Folgenden ist der Inhalt der mit der Dokumentation abgegebenen CD-ROM beschrieben.

- **Dokumentation:** Unter der Dokumentation sind alle wichtigen Dokumente abgelegt, die im Zusammenhang mit der Diplomarbeit entstanden sind.
  - Aufgabenstellung Bachelor Thesis
  - Bericht Bachelor Thesis
- **Code:** Im Verzeichnis Code sind die Artefakte aus dem Softwareentwicklungsprozess abgelegt.
  - Java Source-Code in ZIP-Dateien
  - Javadoc
  - Kompilierte JAR-Archive
- **Workspace:** Unter Workspace ist der gesamte Netbeans-Workspace beider Applikationen sowie der Case Study zur Kanalsicherung abgelegt.

Kapitel 2

Das TrustVote-Protokoll

2.1 Hintergrund

Um dem Leser den Kontext zu verdeutlichen, in welchem die in diesem Bericht beschriebenen Applikationen implementiert wurden, möchten wir an dieser Stelle das TrustVote-Protokoll kurz näher erläutern. Das im Rahmen des Forschungsprojekts TrustVote an der Berner Fachhochschule entwickelte E-Voting Protokoll versteht sich als Erweiterung zu bestehenden, papierbasierten Wahlsystemen. Es baut auf einer bereits vorhandenen, vertrauenswürdigen Infrastruktur auf und soll den Stimmenden ausserdem immer die Möglichkeit einräumen, ihre über E-Voting abgegebene Stimme mit einer Stimmabgabe an der Urne zu widerrufen. Insbesondere soll das Protokoll auch die Forderung nach mehr Offenheit und Transparenz im Bereich E-Voting erfüllen.

2.2 Rollen

Im Folgenden werden die im Protokoll involvierten Komponenten und ihre Rollen näher beschrieben:

- **Die Stimmenden:** Ein Stimmender möchte per E-Voting an einer Wahl oder Abstimmung teilnehmen. Dazu wird er die Voter Application benutzen.
- **Die Administration:** Die Administration initiiert eine Abstimmung oder Wahl. Sie erstellt den elektronischen Wahlzettel und publiziert eine Liste von autorisierten Stimmenden.
- **Die Authorities:** Die Authorities sollen sicherstellen, dass ein Stimmender zur Stimmabgabe autorisiert ist und nicht mehrere Male abstimmen kann. Diese Verantwortung soll auf mehrere, unabhängige Authorities aufgeteilt werden.
- **Die Key Collectors:** Die Key Collectors sind für die Aufbewahrung der Teilschlüssel während der Stimmabgabe verantwortlich. Auch diese Verantwortung wird auf mehrere, unabhängige Instanzen verteilt.

- **Das Public Board:** Auf dem Public Board werden die abgegebenen Stimmen verschlüsselt publiziert. Alle Stimmenden sind berechtigt, Stimmen auf dem Public Board zu lesen. Jedoch ist dies erst möglich, wenn die Teilschlüssel zu einer Stimme von den Key Collectors freigegeben wurden.

- **Der Tallier:** Der Tallier entschlüsselt und zählt am Ende einer Wahl oder Abstimmung alle abgegebenen Stimmen, die genügend gültige Signaturen aufweisen.

2.3 Ablauf

Der Ablauf des TrustVote-Protokolls ist in Abbildung 2.1 dargestellt. Er lässt sich grob in fünf Phasen unterteilen.

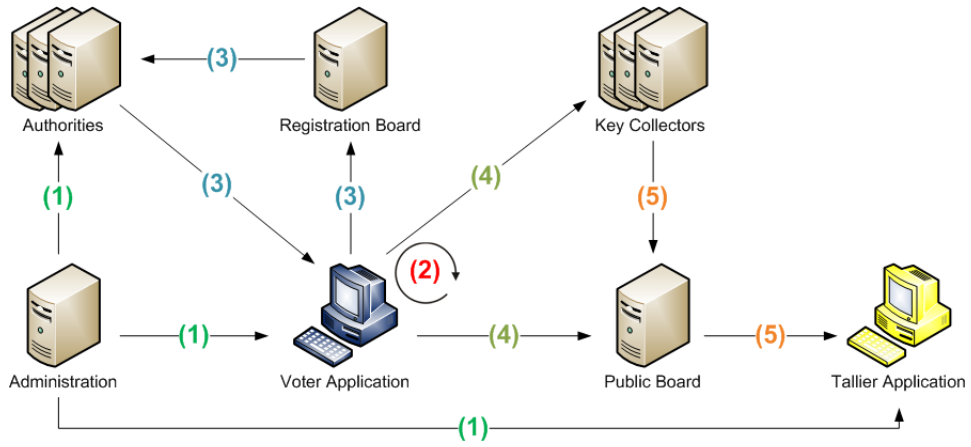

Abbildung 2.1: Ablauf TrustVote-Protokoll

Vorbereitung (1) Die Administration initiiert einen Wahlgang und publiziert einen leeren Stimmzettel sowie die Event Nummer. Ausserdem wird den Authorities mitgeteilt, welche Personen für diesen Wahlgang stimmberechtigt sind.

Erstellen des Stimm-Geheimnisses und des Stimm-Kennwerts (2) Nachdem ein Stimmender den erhaltenen Wahlzettel ausgefüllt hat, wird seine Stimme verschlüsselt. Zudem wird eine Stimm-Kennzahl generiert, welche später der Identifikation der abgegebenen Stimme dient.

Registrierung (3) Die Registrierung stellt sicher, dass ein Stimmender jeweils nur einmal seine Stimme abgeben kann und überhaupt zur Stimmabgabe berechtigt ist. Aus der generierten Stimm-Kennzahl, der Identifikationsnummer des Wahlgangs und der verschlüsselten Stimme wird eine Prüfsumme generiert, welche von der Voter Applikation verblindet und anschliessend von den Authorities signiert wird.

Verarbeitung der Stimme (4) Anschliessend wird die verschlüsselte Stimme zusammen mit den erhaltenen Signaturen der Authorities anonym auf dem Public Board publiziert. Ein Stimmender kann anhand seiner Stimm-Kennzahl überprüfen, ob seine Stimme auf dem Public Board veröffentlicht wurde. Gleichzeitig wird der geheime Schlüssel, welcher zur Verschlüsselung der Stimme verwendet wurde, in n Teilschlüssel aufgeteilt, wobei n die Anzahl Key Collectors ist. Jeder Teilschlüssel wird nun an einen Key Collector versendet.

Zählen (5) Sobald ein Wahlgang geschlossen ist, senden die Key Collectors ihre Teilschlüssel an das Public Board, welches die Teilschlüssel der entsprechenden Stimme zuordnet. Der Tallier kann nun, indem er überprüft, ob genügend gültige Signaturen vorhanden sind, die kompletten Stimmen entschlüsseln und den Wahlgang bzw. die Abstimmung auszählen.

Kapitel 3

Analyse

3.1 Ausgangslage

Da das TrustVote-Protokoll absolut neu ist, gibt es keine bestehenden Lösungen für die benötigten Komponenten. Die Voter und Tallier Anwendungen sollen alle notwendigen Funktionen implmentieren, um ein ordnungsgemässes Funkionieren des Protokolls zu garantieren. Teilweise kann auf bereits bestehende Implementationen im Java Security Provider zurückgegriffen werden. Funktionen wie das (t, N)-Threshold Blind Signature und das Shamir Secret Sharing Verfahren müssen jedoch neu implementiert werden. Zudem sollen die Anwendungen, aufgrund der Demonstration des TrustVote-Protokolls im Rahmen des Swiss E-Voting Workshop 2009, eine ansprechende Benutzeroberfläche aufweisen.

3.2 Projektziele

Ziel ist es mit Hilfe der Voter und Tallier Applikationen das TrustVote-Protokoll im Rahmen des Swiss E-Voting Workshop 2009 vorzustellen. Die Komponenten sollen soweit umgesetzt und in das Protokoll integriert werden, dass eine Wahl oder Abstimmung zu Demonstrationszwecken durchgeführt werden kann. Ausserdem soll die Benutzeroberfläche einfach, aber ansprechend für die Teilnehmer des Swiss E-Voting Workshops 2009 gestaltet werden. Den folgenden primären Anforderungen sollten die zu realisierenden Anwendungen gerecht werden:

- Ein Stimmender kann einen leeren Wahlzettel ausfüllen und anschliessend versenden, so dass dieser korrekt vom Public Board weiterverarbeitet sowie von der Tallier Applikation gezählt werden kann.
- Ein Stimmender kann mit Hilfe der Tallier Applikation die per Voter Applikation abgegebenen Stimmen abholen und zählen lassen. Die Resultate sollen dem Benutzer ansprechend, vorzugsweise mit Diagrammen, präsentiert werden.
- Die Benutzeroberflächen beider Applikationen sollen für einen ungeübten Benutzer leicht bedienbar sein. Ausserdem ist sicherzustellen, dass diese den Anforderungen an eine öffenliche Demonstration gerecht werden.

Die Anwendungen sollten überdies bei genügend zeitlichen Ressourcen folgende sekundären Anforderungen implementieren:

- Da der Swiss E-Voting Workshop 2009 zweisprachig, in französischer und deutscher Sprache, durchgeführt wird, sollten auch beide Applikationen in den erwähnten Sprachen verfügbar sein. Dieses Ziel wurde erst während der Phase der Realisierung definiert.
- Die im Protokoll vorgesehene Sicherung der Übertragungskanäle soll implementiert werden. Dies hat jedoch für die Demonstration am Swiss E-Voting Workshop 2009 eine geringe Priorität.

3.3 Projektrisiken

Die Entwicklung eines Systems beinhaltet immer gewisse technologische, quantitative und personelle Risiken. Im Rahmen einer Analyse wurde versucht, die Risiken zu identifizieren und diese mit geeigneten Massnahmen zu reduzieren.

Identifizierte Risiken:

- Es werden Technologien eingesetzt, über welche wenig Erfahrung vorhanden ist (Java, GUI Programmierung).
- Das Projekt ist abhängig von der Realisierung der restlichen Komponenten des TrustVote-Protokolls.
- Es werden kryptographische Verfahren eingesetzt, zu welchen noch keine oder wenig Erfahrung vorhanden ist (Shamir Secret Sharing Schema, (t, N)-Threshold Blind Signature Schema).

Massnahmen:

- Das technologische Risiko soll durch Einarbeiten in die Thematik sowie durch Realisierung von Prototypen minimiert werden.
- Um den Administrationsaufwand zu verringern, sollen möglichst viele Komponenten unabhängig von den restlichen Anwendungen im TrustVote-Protokoll umgesetzt werden. Es soll genügend Zeit für die Integration der Voter und Tallier Anwendungen in das Gesamtsystem eingeplant werden.
- Um sich in die zu verwendenden kryptographischen Verfahren einarbeiten zu können, soll am Anfang des Projekts genügend Zeit für die Informationsbeschaffung eingeplant werden.

3.4 Umfeld und Rahmenbedingungen

Das Projekt findet im Rahmen der Umsetzung des an der Berner Fachhochschule entwickelten TrustVote-Protokolls statt. Die beiden zu realisierenden Applikationen

sollen zusammen mit den restlichen Komponenten, welche es umzusetzen gilt, am Swiss E-Voting Workshop 2009 die Funktionsweise des TrustVote-Protokolls verdeutlichen. Die Umsetzung der Voter und Tallier Applikationen findet im Kontext des TrustVote Projekts statt. Eine enge Zusammenarbeit mit den Mitarbeitern des Projekt ist für eine erfolgreiche Umsetzung des TrustVote-Protokolls notwendig.

3.5 Abgrenzungen

Mit dem Projekt wird speziell auf die Demonstration am Swiss E-Voting Workshop 2009 hingearbeitet. Im Bereich der Benutzeroberfläche wird teilweise bewusst auf Flexibilität verzichtet, da immer der selbe Stimmzettel verwendet wird. Die zu implementierenden Sprachen sollen nicht durch den Benutzer wählbar sein. Auch wird einer einfachen Installation und dem Zugang zu den Anwendungen wenig Beachtung geschenkt. Die Arbeit soll sich auf die technologische Umsetzung der Projektziele konzentrieren und weniger auf die einfache Installation und den Zugang zu den Anwendungen für potentielle Benutzer.

3.6 Erweiterungen

Sämtliche unter dem Titel “Abgrenzungen” aufgeführten Punkte könnten Teil einer Erweiterung werden. Ausserdem wäre in Betracht zu ziehen, die Voter und Tallier Anwendungen als Webapplikationen umzusetzen, was eine lokale Installation bei den Benutzern überflüssig machen würde.

3.7 Funktionale Anforderungen

Im Folgenden findet sich eine kurze Übersicht über die definierten Use-Cases für Voter und Tallier Applikationen. Die detaillierte Beschreibung ist im Anhang A zu finden.

3.7.1 Rollen (Actors)

Tabelle 3.1 vermittelt eine Übersicht über die Rollen der Voter und Tallier Applikationen.

Actor	Beschreibung
Voter	Füllt einen Stimmzettel aus und versendet diesen
Tallier	Zählt die Stimmen und stellt die Resultate dar
Administration	Administriert einen Voting Event
Authority	Stellt sicher, dass ein Voter zur Stimme berechtigt ist
Registration Board	Nimmt die Registrierung eines Voters entgegen
Public Board	Nimmt den ausgefüllten Stimmzettel des Voters entgegen
Key Collector	Nimmt den Teilschlüssel des Voters entgegen

Tabelle 3.1: Rollen (Actors)

3.7.2 Voter Applikation

Die Use-Cases der Voter Applikation sind in Abbildung 3.1 dargestellt.

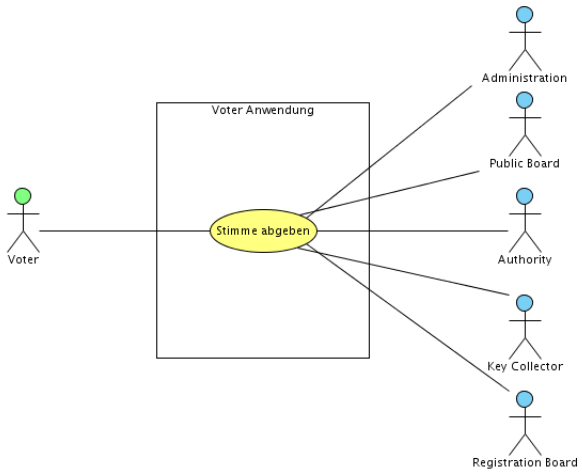

Abbildung 3.1: UC1: Voter Anwendung

3.7.3 Tallier Applikation

Die Use-Cases der Tallier Applikation sind in Abbildung 3.2 dargestellt.

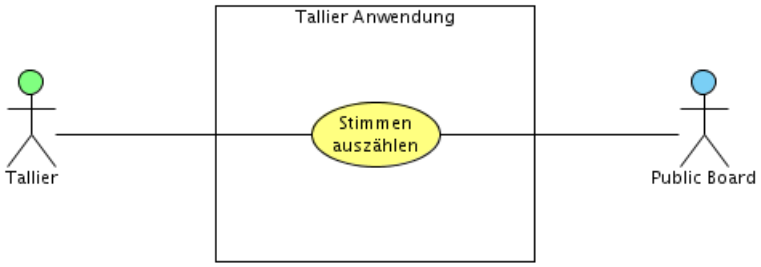

Abbildung 3.2: UC2: Tallier Anwendung

3.8 Technische Anforderungen

Da die Komponenten des TrustVote-Protokolls alle mit der Programmiersprache Java implementiert werden, sollen auch die Voter und Tallier Applikation mit Java umgesetzt werden. Für die Umsetzung der verschiedenen kryptographischen Funktionen soll daher die Java Cryptography Architecture (JCA) 6[1] verwendet werden.

Aufgrund des Designs des TrustVote-Protokolls ist es notwendig, dass die kryptographischen Verfahren Shamir Secret Sharing[4] und Threshold Blind Signature[3] für das Erstellen der Teilschlüssel und das Signieren der Stimmen verwendet werden. Ausserdem muss beachtet werden, dass die folgenden, per Definition von den Datentypen des Systems unterstützten, Verschlüsselungs-, Signatur- und Hashalgorithmen

[1] Java Cryptography Architecture (JCA) ist eine standardisierte Java-Programmierschnittstelle für Kryptographie, http://java.sun.com/javase/6/docs/technotes/guides/security/crypto/CryptoSpec.html

verwendet werden, wobei nicht alle unterstützten Algorithmen implementiert werden müssen.

- Advanced Encryption Standard (AES)
- Triple Data Encryption Standard (3DES)
- RSA Algorithmus
- Digital Signature Algorithm (DSA)
- Secure Hash Algorithm (SHA)

3.9 Nichtfunktionale Anforderungen

In diesem Abschnitt werden die Anforderungen beschrieben, welche die Funktionalität des Gesamtsystems nur indirekt beeinflussen.

3.9.1 Mehrsprachigkeit

Die Benutzeroberfläche der Voter und Tallier Applikationen soll Mehrsprachigkeit unterstützen. Mit Sicht auf die Präsentation am Swiss E-Voting Workshop sollen die Sprachen Deutsch und Französich implementiert werden. Es ist jedoch keine Anforderung, dass die Benutzer der Anwendungen die Sprache selbst auswählen können.

Kapitel 4

Design

4.1 Allgemein

Wie ein E-Voting-Protokoll selbst, müssen auch die implementierenden Applikationen vertrauenswürdig aufgebaut sein. Das heisst, dass das Design so einfach wie möglich gehalten werden sollte, damit die Relationen zwischen den Klassen und der eigentliche Ablauf des Programms klar verständlich sind. Daher wird das Design nach dem **K**eep **I**t **S**imple and **S**mart (kurz KISS) Prinzip aufgebaut. Auf Komponenten wie Interfaces, abstrakte Klassen oder Design Patterns wird bewusst verzichtet. Nur die notwendigsten Klassen werden implementiert, um die Transparenz so hoch wie möglich zu halten.

4.1.1 Vordefinierte Komponenten

Common-Datatypes

Die Common-Datatypes bilden das Datenfundament der Voter- und Tallier-Applikation. Sämtliche relevanten Informationen für eine Stimmabgabe sowie deren Auszählung sind entweder in diesen Daten enthalten oder werden darin abgespeichert. Bereits im Vorfeld der Projektrealisierung wurden die Common-Datatypes definiert und im XML-Schema `Datatypes.xsd` abgelegt. Mit Hilfe von JAXB werden sie an Java-Klassen gebunden, damit die Datentypen verwendet werden können. In der Abbildung 4.1 befindet sich eine Übersicht der Java-Objekte. Bei einigen Klassen wurde der Methodenblock weggelassen, da diese selbsterklärend sind.

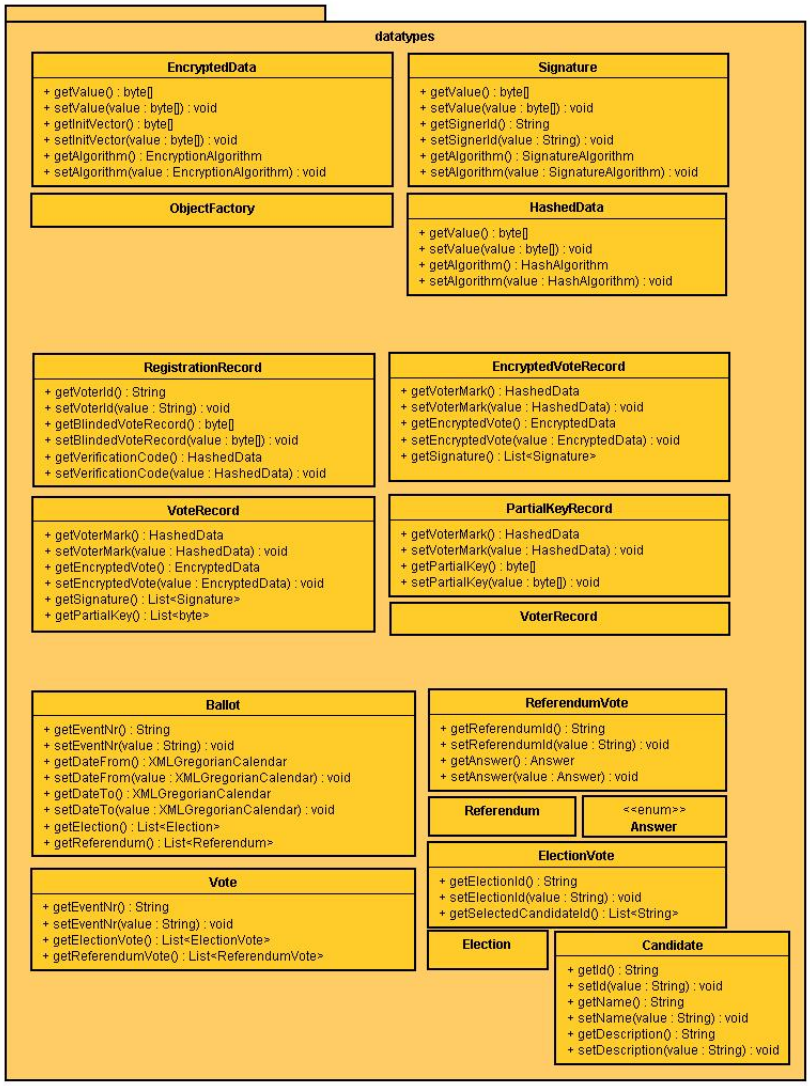

Abbildung 4.1: Übersicht der Datentypen

Klasse	Beschreibung
EncryptedData	Enthält verschlüsselte Daten sowie den verwendeten Verschlüsselungsalgorithmus und Initialvektor IV.
Signature	Enthält Signaturen mit dem verwendeten Algorithmus und der Identität des Signierers.
ObjectFactory	Eine Objektfabrik, die sämtliche hier aufgelisteten Datentypen erstellen kann.
HashedData	Enthält gehashte Daten mit dem verwendeten Hash-Algorithmus.
RegistrationRecord	Enthält die Voter-ID, die zu signierenden, verblindeten Daten und den Verifizierungscode. Der RegistrationRecord wird auf dem Registration Board abgelegt und soll berechtige Stimmpersonen authorisieren sowie Doppelstimmen verhindern.
EncryptedVoteRecord	Enthält den Stimm-Kennwert (VoterMark) der Stimmperson, die verschlüsselte Stimme und eine Liste mit Signaturen. Der EncryptedVoteRecord wird in dieser Form auf dem Public Board abgelegt.
VoteRecord	Enthält den Stimm-Kennwert (VoterMark) der Stimmperson, die verschlüsselte Stimme, eine Liste mit Signaturen und eine Liste mit Teilschlüsseln. Der VoteRecord wird am Ende des Abstimmungsevents dem Zähler übergeben.
VoterRecord	Enthält die Voter-ID und das gehashte Credential. Der VoterRecord wird den Authorities gesendet. Diese speichern die Daten ab und erstellen so eine neue stimmberechtigte Person in ihrer Datenbank.
PartialKeyRecord	Enthält den Stimm-Kennwert (VoterMark) der Stimmperson sowie einen Teilschlüssel des geheimen Schlüssels, mit dem die Stimme geschützt wurde. Die PartialKeyRecords werden den Key Collectors zugestellt.
Ballot	Enhält alle notwendigen Daten des Abstimmungszettels, wie die Abstimmungsnummer (EventNr), das Start- und Enddatum und, falls vorhanden, anstehende Wahlen und/oder Referenden.
Vote	Enthält die Abstimmungsnummer (EventNr) und die abgegebenen Stimmen für vorhandene Wahlen und/ oder Referenden.
Referendum	Enthält eine eindeutige ID und ein Referendumstext.
ReferendumVote	Enthält die Antwort für ein spezifisches Referendum sowie die ID dieses Referendums.
Answer	Enthält die möglichen Antworten auf ein Referendum: Ja, Nein oder Enthaltung.
Election	Enthält eine eindeutige ID, eine Liste mit Kandidaten, eine Beschreibung der Wahl und eine Mindest- und Maximalanzahl der erlaubten Stimmen.
Candidate	Enthält eine eindeutige ID, den Namen des Kandidaten sowie dessen Beschreibung.

Tabelle 4.1: Klassenbeschreibung des Pakets datatypes

Common-Configuration

Eine weitere, bereits vordefinierte Komponente ist die Common-Configuration. Die Voter Applikation ist damit für den Administrator bis zu einem gewissen Grad konfigurierbar. Einerseits werden darin die Web Services definiert und andererseits die kryptographischen Algorithmen für Verschlüsselung, Hash-Codierung und Signierung. Die Informationen sind wie bei den Common-Datatypes in einem XML-Schmema `configuration.xml` gespeichert, die mit JAXB in Java-Klassen konvertiert werden. Um die Konfiguration verwenden zu können, muss die betreffende Java-Klasse ein neues ConfigurationContext-Objekt erstellen (siehe Abbildung 4.2). Dadurch wird der Inhalt der XML-Datei verfügbar. Ein Webservice wird durch folgende Tag-Elemente in der XML-Datei definiert:

Tag Typ	**Tag**	**Beschreibung**
Parent-Tag	<Name>	Name des Service und gleichzeitig Namensgeber für das Java-Objekt. Beispielsweise wird ein <Administrator> zu einem Administrator-Objekt.
Child-Tag	<id>	Eindeutige Identifikator des Web Services.
Child-Tag	<url>	URL des Web Services.

Tabelle 4.2: Web Service-Definition in configuration.xml

Ein kryptographischer Algorithmus wird durch folgende Tag-Elemente definiert:

Tag Typ	**Tag**	**Beschreibung**
Child-Tag	<signatureAlgorithm>	Algorithmus für die Erstellung und Verfikiation von Signaturen.
Child-Tag	<encryptionAlgorithm>	Algorithmus für die Ver- und Entschlüsselung von Daten.
Child-Tag	<hashAlgorithm>	Algorithmus für die Hash-Codierung von Daten.

Tabelle 4.3: Web Service-Definition in configuration.xml

Im Codelisting 4.1 ist der Inhalt der Datei configuration.xml abgebildet.

```
<?xml version="1.0" encoding="UTF-8"?>
<ns1:configuration  xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
    xmlns:ns1='http://showcase.trustvote.bfh.ch/configuration'
    xsi:schemaLocation='http://showcase.trustvote.bfh.ch/configuration file:/
        Users/ara/NetBeansProjects/common-config/xml-resources/jaxb/Config/
        Configuration.xsd'>
     <entities>
         <administration>
             <id>admin1</id>
             <url>http://vmnoah.bfh.ch:8080/AdministrationService/
                 AdministratorService</url>
         </administration>
         <authority>
             <id>auth1</id>
             <url>http://vmliza.bfh.ch:8160/AuthorityService/AuthorityService</
                 url>
         </authority>
         <authority>
             <id>auth2</id>
             <url>http://vmliza.bfh.ch:8261/AuthorityService/AuthorityService</
                 url>
         </authority>
         <authority>
             <id>auth3</id>
             <url>http://vmliza.bfh.ch:8362/AuthorityService/AuthorityService</
                 url>
         </authority>
         <authority>
             <id>auth4</id>
             <url>http://vmliza.bfh.ch:8463/AuthorityService/AuthorityService</
                 url>
         </authority>
         <authority>
             <id>auth5</id>
             <url>http://vmliza.bfh.ch:8564/AuthorityService/AuthorityService</
                 url>
         </authority>
         <collector>
             <id>coll1</id>
             <url>http://vmaura.bfh.ch:8160/CollectorService/CollectorService</
                 url>
         </collector>
         <collector>
             <id>coll2</id>
             <url>http://vmaura.bfh.ch:8261/CollectorService/CollectorService</
                 url>
         </collector>
         <collector>
             <id>coll3</id>
             <url>http://vmaura.bfh.ch:8362/CollectorService/CollectorService</
                 url>
         </collector>
         <collector>
             <id>coll4</id>
             <url>http://vmaura.bfh.ch:8463/CollectorService/CollectorService</
                 url>
         </collector>
         <collector>
             <id>coll5</id>
             <url>http://vmaura.bfh.ch:8564/CollectorService/CollectorService</
                 url>
         </collector>
```

```
50          <registrationBoard>
51              <id>regb1</id>
52              <url>http://vmnoah.bfh.ch:8080/RegistrationBoardService/
                    RegistrationBoardService</url>
53          </registrationBoard>
54          <votingBoard>
55              <id>votb1</id>
56              <url>http://vmnoah.bfh.ch:8080/VotingBoardService/
                    VotingBoardService</url>
57          </votingBoard>
58      </entities>
59      <algorithms>
60          <signatureAlgorithm>RSA</signatureAlgorithm>
61          <encryptionAlgorithm>AES256</encryptionAlgorithm>
62          <hashAlgorithm>SHA1</hashAlgorithm>
63      </algorithms>
64  </ns1:configuration>
```

Listing 4.1: Konfigurationsdatei Voter

Die generierten Java-Klassen von `configuration.xml` sind in der Abbildung 4.2 als Klassendiagramm dargestellt.

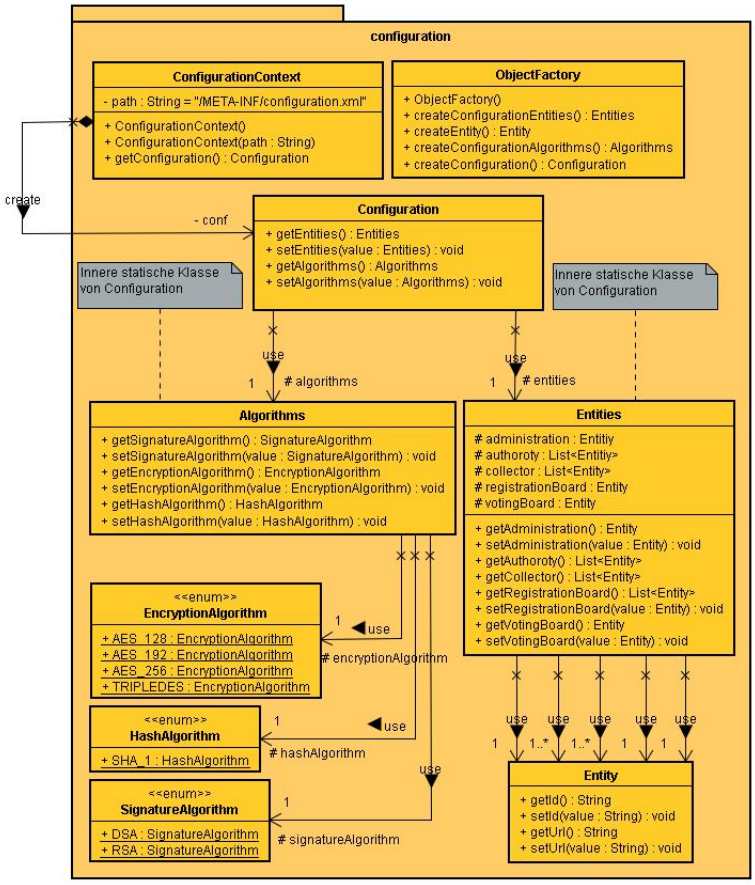

Abbildung 4.2: Common-Configuration in Java

Klasse	Beschreibung
ObjectFactory	Liefert neue Algorithmus- und Entitäts-Objekte, die innerhalb und ausserhalb der Configuration-Klasse verwendet werden können.
ConfigurationContext	Generiert ein Configuration-Objekt, das als Inhalt die Daten des `configuration.xml` gespeichert hat. Es ist auch möglich, eine selbst erstellte Konfigurationsdatei anzugeben.
Configuration	Stellt die verfügbaren Algorithmen und Entitäten bereit.
Algorithms	Stellt die verfügbaren Signierungs-, Verschlüsselungs- und Hash-Algorithem bereit.
EncryptionAlgorithm	Beinhaltet die erlaubten Verschlüsselungsalgorithmen AES_128, AES_192, AES_256 und Triple-DES, die zuvor von einer berechtigten Person festgelegt wurden. Weitere Algorithmen sind nicht zugelassen.
HashAlgorithm	Beinhaltet den erlaubten Hashalgorithmus SHA-1, der zuvor von einer berechtigten Person festgelegt wurde. Weitere Algorithmen sind nicht zugelassen.
SignatureAlgorithm	Beinhaltet die erlaubten Signierungsalgorithmen DSA und RSA, die zuvor von einer berechtigten Person festgelegt wurden. Weiter Algorithmen sind nicht zugelassen.
Entities	Stellt die Administrations-, Authorities-, Collector-, RegistrationBoard- und VotingBoard-Entitäten aus der `configuration.xml` Datei bereit. Akzeptiert auch neue Entity-Objekte für die Web Services.
Entity	Stellt die Web Service-Attribute `id` und `url` zur Verfügung. Akzeptiert auch neue Entity-Objekte.

Tabelle 4.4: Klassenbeschreibung des Pakets configuration

Web Services

Wo die Informationen zu den Web Services abgelegt sind und wie man sie in Java verwenden kann, wurde bereits im Kapitel 4.1.1 erwähnt. Zusätzlich ist nun eine Schnittstelle zu den Services nötig, um Daten hin und her schicken zu können. Dabei ist `JAX-WS` behilflich, eine API zum Erstellen von Web Services, der man als Quelle die `WDSL-Datei` des spezifischen Web Services angibt. Daraus entstehen Java-Klassen. Die Abbildung 4.3 bietet eine Übersicht der erstellten Klassen und die darauf folgende Tabelle liefert die Beschreibung dazu. Da die Klassen vom Gebrauch her äquivalent sind, ist als Beispiel eine Beschreibung ausreichend. Das Listing 4.2 zeigt die WSDL-Datei des `AdministrationService`.
Die Verbindung zwischen Common-Config und den Web Services besteht darin, dass mit Hilfe der `URL` aus der Konfiguration, angewendet auf die Web Service-Objekte, jeder Service eines Typs durch Iteration verwendbar ist. Bei jedem Durchlauf wird der aufrufenden Methode eine andere URL übergeben, um so beispielsweise von jeder Authority die Signatur zu erhalten.

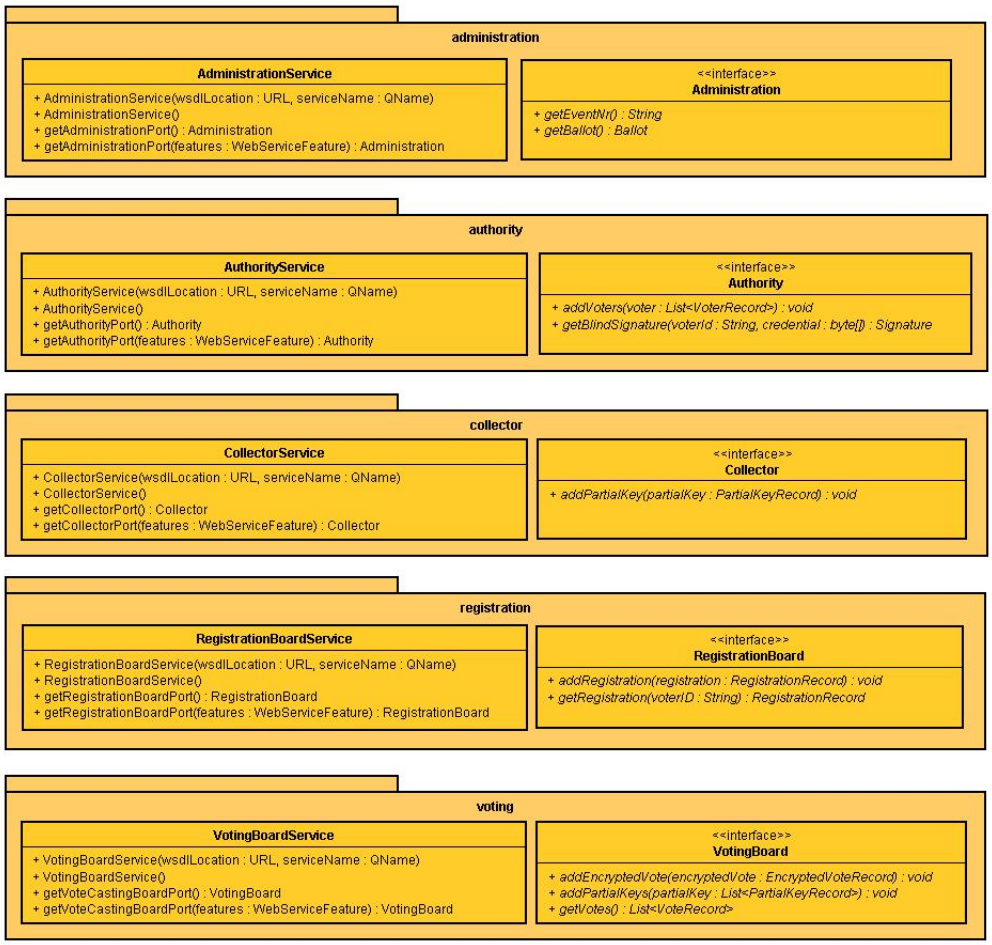

Abbildung 4.3: Klassendiagramm Web Services

Paket	Klasse	Beschreibung
administration	AdministrationService	Beinhaltet die URL der WSDL-Datei, den Servicenamen und den nötigen Port, um mit dem Service kommunizieren zu können. Es kann auch ein Objekt erzeugt werden, wo diese beiden Attribute angegeben werden können. Die Methoden `getAdministrationPort` liefern eine Administraton Instanz.
	Administration	Mit dem Interface Administration können die Methoden `getEventNr` und `getBallot` des Web Service aufgerufen werden.

Tabelle 4.5: Klassenbeschreibung eines Web Service Pakets

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2
3  <!-- Authors: Stephan Fischli, Rolf Haenni -->
4
5  <definitions name="AdministrationService" xmlns="http://schemas.xmlsoap.org/
       wsdl/"
6    targetNamespace="http://showcase.trustvote.bfh.ch/administration"
7    xmlns:tns="http://showcase.trustvote.bfh.ch/administration"
8    xmlns:msg="http://showcase.trustvote.bfh.ch/administration/messages"
9    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
10   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
11
12   <types>
13     <xsd:schema>
14       <xsd:import namespace="http://showcase.trustvote.bfh.ch/administration/
             messages"
15         schemaLocation="AdministrationService.xsd"/>
16     </xsd:schema>
17   </types>
18
19   <message name="getEventNr">
20     <part name="parameters" element="msg:getEventNr"/>
21   </message>
22   <message name="getEventNrResponse">
23     <part name="parameters" element="msg:getEventNrResponse"/>
24   </message>
25   <message name="getBallot">
26     <part name="parameters" element="msg:getBallot"/>
27   </message>
28   <message name="getBallotResponse">
29     <part name="parameters" element="msg:getBallotResponse"/>
30   </message>
31   <message name="EventNotStartedException">
32     <part name="fault" element="msg:faultInfo"/>
33   </message>
34
35   <portType name="Administration">
36     <operation name="getEventNr">
37       <input message="tns:getEventNr"/>
38       <output message="tns:getEventNrResponse"/>
39       <fault message="tns:EventNotStartedException" name="
             EventNotStartedException"/>
40     </operation>
41     <operation name="getBallot">
42       <input message="tns:getBallot"/>
43       <output message="tns:getBallotResponse"/>
44       <fault message="tns:EventNotStartedException" name="
             EventNotStartedException"/>
45     </operation>
46   </portType>
47
48   <binding name="AdministrationBinding" type="tns:Administration">
49     <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="
           document"/>
50     <operation name="getEventNr">
51       <soap:operation soapAction=""/>
52       <input>
53         <soap:body use="literal"/>
54       </input>
55       <output>
56         <soap:body use="literal"/>
57       </output>
58       <fault name="EventNotStartedException">
```

```
59          <soap:fault name="EventNotStartedException" use="literal"/>
60        </fault>
61      </operation>
62      <operation name="getBallot">
63        <soap:operation soapAction=""/>
64        <input>
65          <soap:body use="literal"/>
66        </input>
67        <output>
68          <soap:body use="literal"/>
69        </output>
70        <fault name="EventNotStartedException">
71          <soap:fault name="EventNotStartedException" use="literal"/>
72        </fault>
73      </operation>
74    </binding>
75
76    <service name="AdministrationService">
77      <port name="AdministrationPort" binding="tns:AdministrationBinding">
78        <soap:address location="REPLACE_WITH_ACTUAL_URL"/>
79      </port>
80    </service>
81
82  </definitions>
```

Listing 4.2: WSDL-Datei des AdministrationService

Java KeyStore

Die Public Keys der Authorities, die im (t, N)-Threshold Blind Signature-Verfahren verwendet werden, sind in einem Java KeyStore abgelegt. Der KeyStore basiert auf einer Datenbank und wird verwendet, um eine Menge von kryptographischen Zertifikaten und Schlüsseln abzulegen. Der Inhalt kann mit einem Passwort geschützt werden, was jedoch für dieses Projekt nicht nötig war, da es sich beim Inhalt um öffentliche Schlüssel handelt, die nicht geheim gehalten werden müssen. In der Voter- und Tallier-Applikation ist der Java KeyStore im Paket `META-INF` zu finden. `META-INF` ist ein optionales Verzeichnis einer `JAR-Datei`. In unserem Fall gehört je ein `META-INF` Verzeichnis zum Voter.jar und zum Tallier.jar. Es dient als Ablageort für Paket-, Konfigurations-, Sicherheits-, Versions-, Erweiterungs- und Servicedaten, welche die JAVA Platform erkennt und interpretieren kann.
Die Klasse `KeyManager.java` liest schlussendlich die Public Keys aus dem Store und speichert sie in Java-Objekte ab.

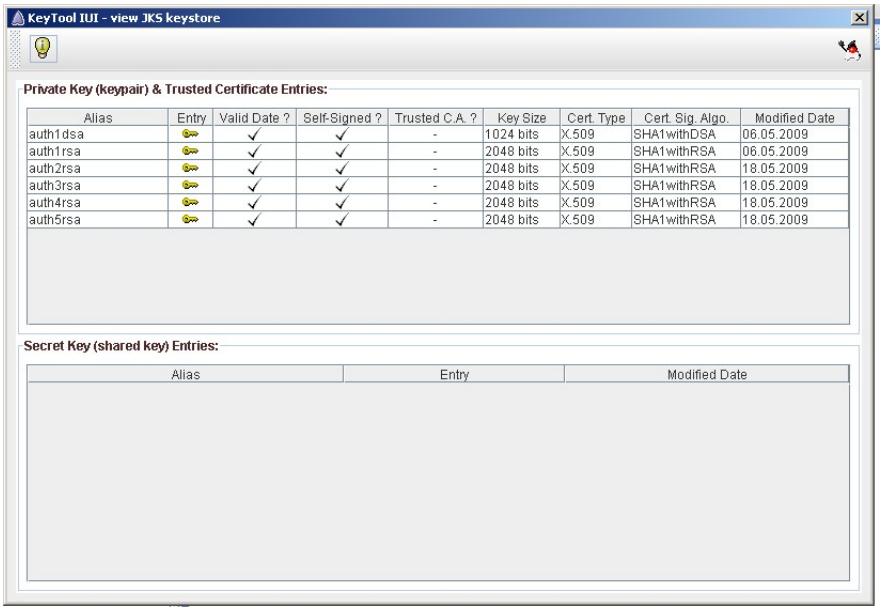

Abbildung 4.4: Public Keys der Autorities im Java KeyStore

4.2 Voter

Mit der Voter Applikation kann eine stimmberechtigte Person an einer elektronischen Wahl oder Abstimmung teilnehmen. Dabei füllt sie mit der grafischen Benutzeroberfläche den Wahlzettel aus und leitet mit *Senden* den eigentlichen Abstimmungsprozess ein. Die folgende Auflistung zeigt die Hierarchie der Applikation anhand von Layers auf. Zu jedem Layer werden die dazugehörigen Pakete dargestellt, welche wiederum die Klassen enthalten, die die Abgabe einer Stimme, von den kryptographischen Verfahren bis zur Platzierung auf dem Public Board, ermöglichen:

- **UI-Layer**
  - gui
- **Application-Layer**
  - controller
- **Domain-Layer**
  - cryptomanager
  - connectionmanager
- **Service-Layer**
  - crypto
  - administration, collector, voting, authoroty, registration
  - configuration
  - commondata
  - datatypes
  - META-INF

4.2.1 Klassenübersicht

Das folgende Klassendiagramm zeigt eine Übersicht aller Klassen und deren Abhängigkeiten auf. Die einzelnen Klassen werden in Anhang B.2 detailliert beschrieben und dargestellt. Die im Diagramm dargestellten hellblauen Pakete mit den violetten Klassen wurden im Rahmen dieses Projekts realisiert. Alle anderen Elemente, welche in einer anderen Farbe dargestellt werden, standen zur Verfügung und mussten nicht selbst implementiert werden.

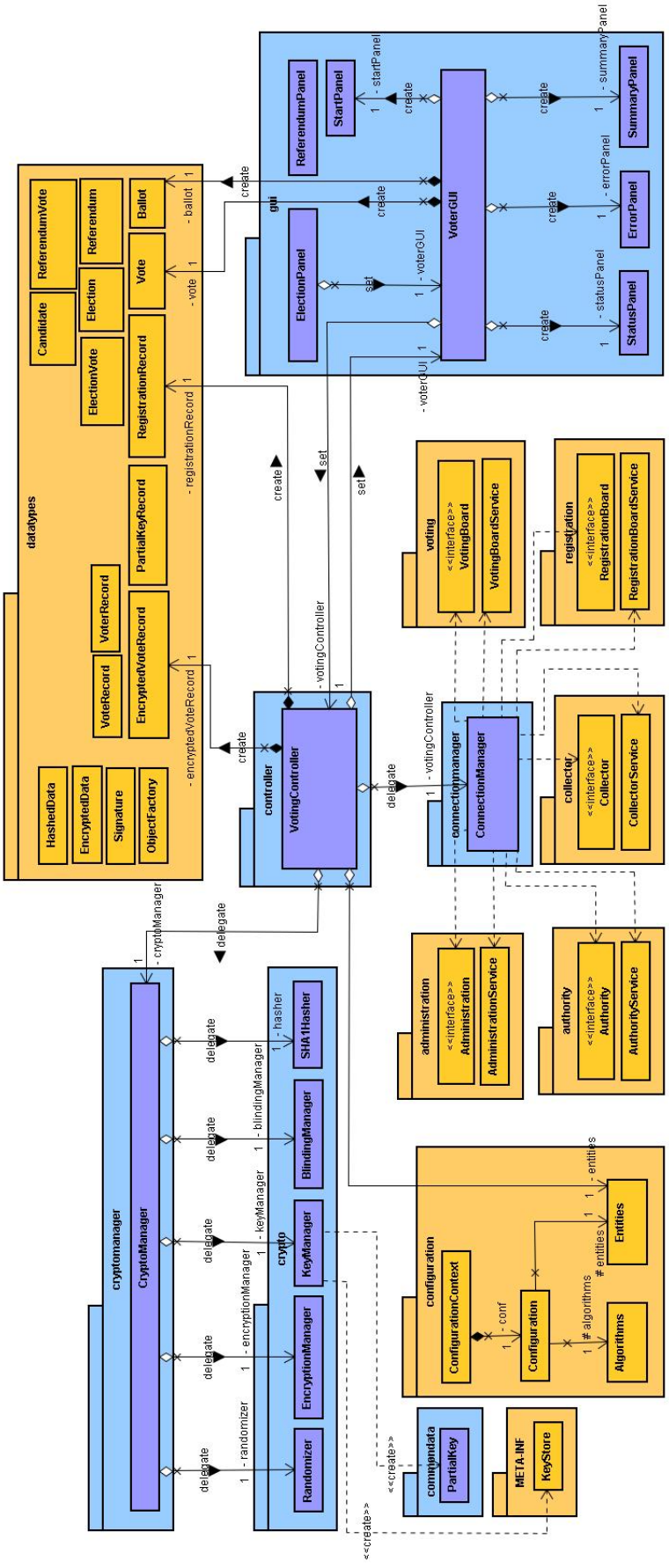

Abbildung 4.5: Übersicht Klassendiagramm des Voters

4.2.2 Ablauf Stimme abgeben

Die Abgabe einer Stimme gliedert sich in folgende Schritte:

1. Die stimmende Person initialisiert mit einem Klick auf den `Senden` Button die `sendVote` Methode im `VotingController`.
2. Die aktuelle Event Nummer wird vom Administration Service geholt.
3. Das Stimm-Geheimnis wird generiert und der stimmenden Person am GUI angezeigt.
4. Durch Hashen des Stimm-Geheimnisses wird der Stimm-Kennwert erstellt.
5. Der geheime Schlüssel wird erzeugt.
6. Die Stimme wird mit dem geheimen Schlüssel verschlüsselt.
7. Die Public Keys der Authorities werden aus dem Java KeyStore extrahiert.
8. Der Verblindungsfaktor wird erstellt.
9. Mit dem Verblindungsfaktor und den Public Keys werden die individuellen Verblindungsfaktoren der Authorities erzeugt.
10. Der Stimm-Kennwert zusammen mit der verschlüsselten Stimme und der Event Nummer werden gehasht.
11. Der eben erstellte Hashwert wird verblindet.
12. Der verblindete Hashwert wird zusammen mit dem Credential gehasht. Es entsteht der Verifizierungscode.
13. Der Registration Record wird aus der Voter ID, dem verblindeten Hashwert und dem Verifizierungscode erstellt.
14. Der Registration Record wird über den ConnectionManager an das Registration Board gesendet.
15. Mit der Voter ID und dem Credential als Identifikation wird von jeder Authority die Signatur geholt.
16. Sämtliche erhaltenen Signaturen werden mit dem individuellen Verblindungsfaktor und dem Public Key der Authority entblindet.
17. Jede entblindete Signatur wird auf ihre Gültigkeit überprüft, indem die gehashten Daten mit dem Public Key der signierenden Authority verifiziert wird.
18. Die Teilschlüssel für die Key Collectors werden aus dem geheimen Schlüssel erstellt.

19. Für jeden Teilschlüssel wird zusammen mit dem Stimm-Kennwert ein PartialKeyRecord erstellt und über den ConnectionManager den Key Collectors hinzugefügt.

20. Schlussendlich wird aus dem Stimm-Kennwert, der verschlüsselten Stimme und den entblindeten Signaturen ein EncryptedVoteRecord erstellt, der über den ConnectionManager an das Voting Board gesendet wird. Somit ist die Stimmabgabe vollzogen.

Die folgenden Sequenzdiagramme 4.6, 4.7, 4.8 und 4.9 zeigen den detaillierten Sequenzablauf und alle beteiligten Klassen des Sendeprozesses.

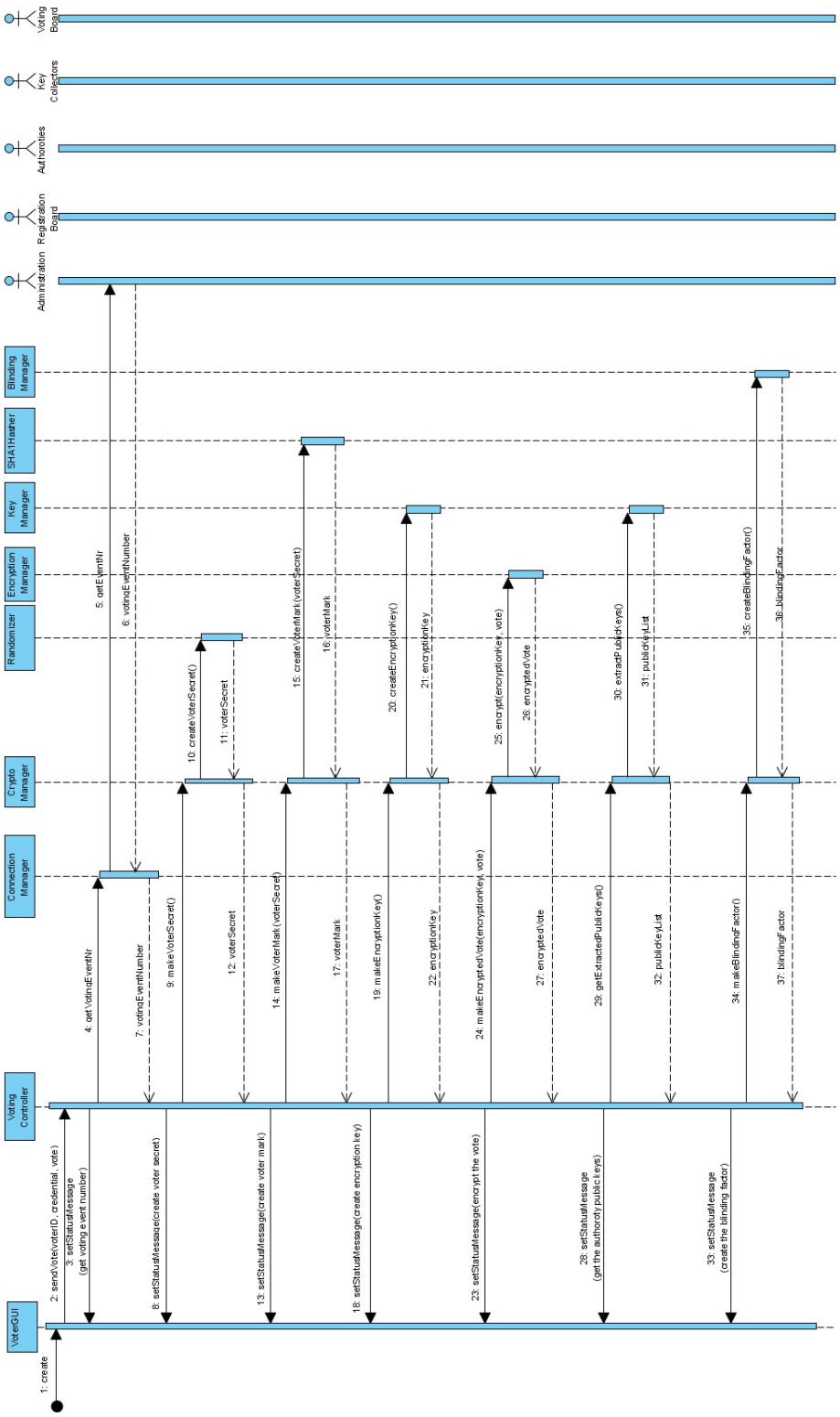

Abbildung 4.6: Sequenzdiagramm Teil 1 der Stimmabgabe

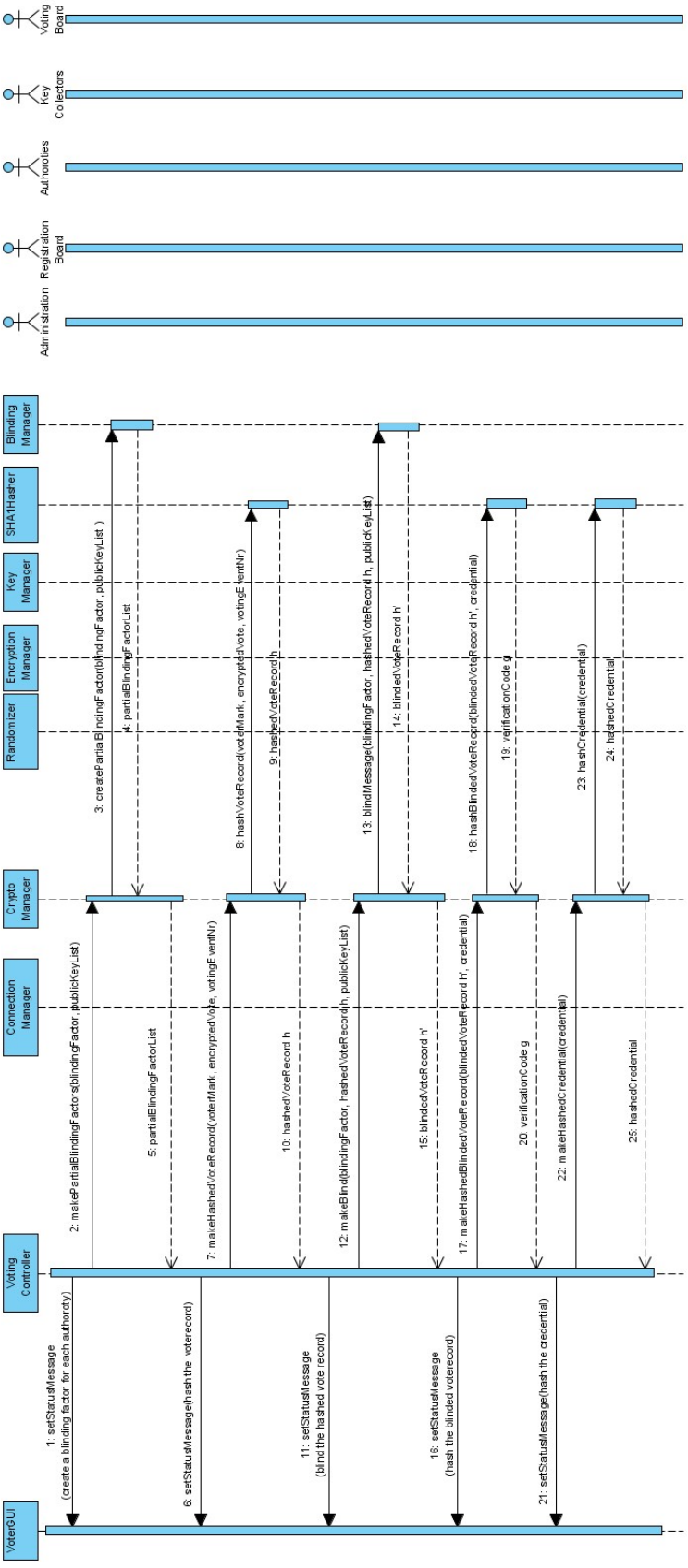

Abbildung 4.7: Sequenzdiagramm Teil 2 der Stimmabgabe

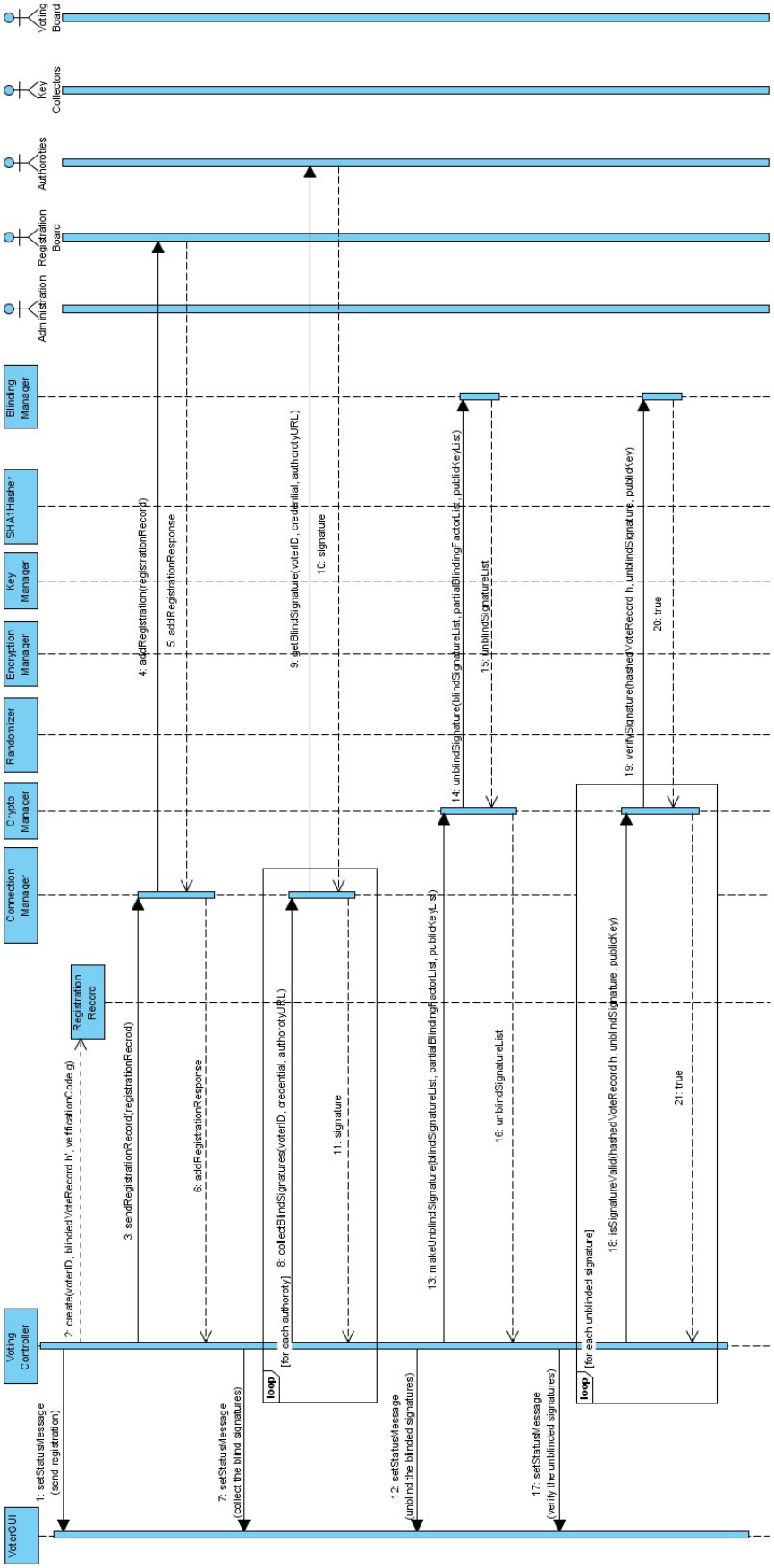

Abbildung 4.8: Sequenzdiagramm Teil 3 der Stimmabgabe

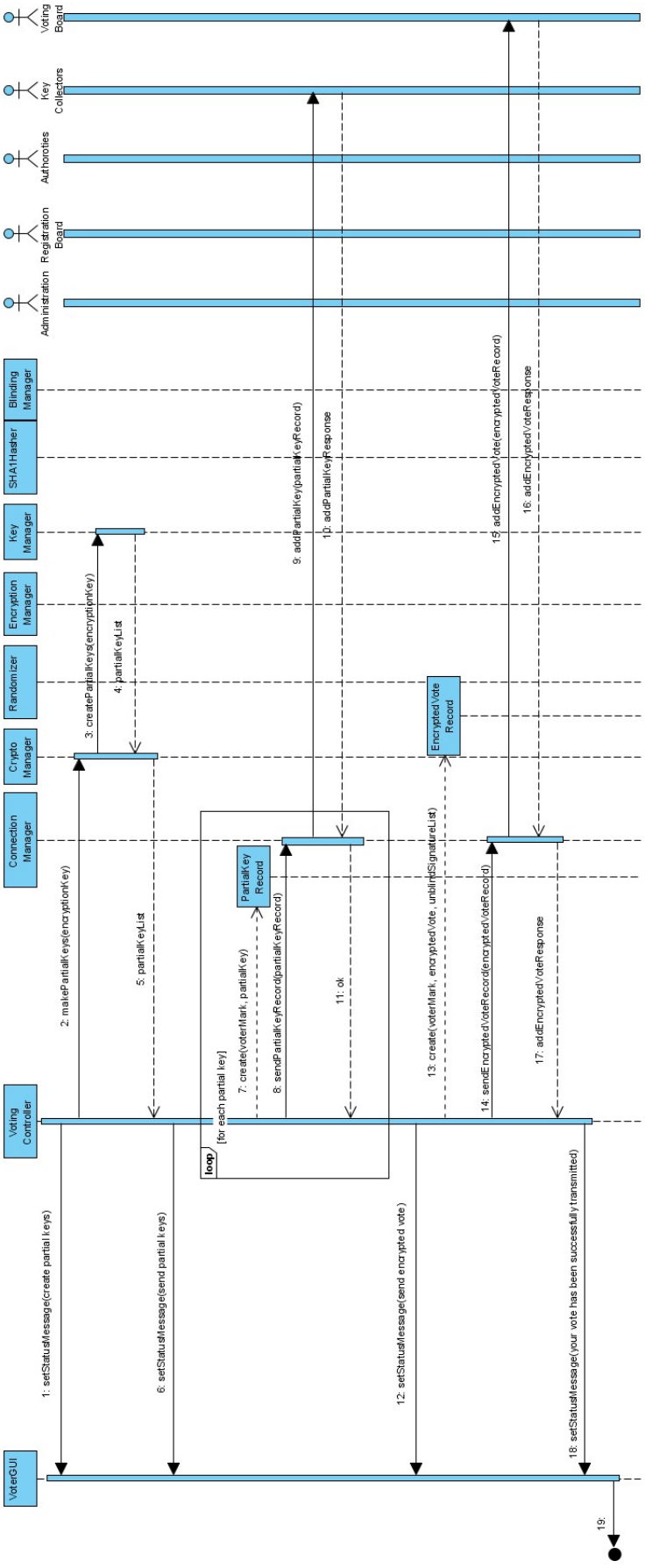

Abbildung 4.9: Sequenzdiagramm Teil 4 der Stimmabgabe

4.3 Tallier

Mit der Tallier Applikation kann eine Partei eine elektronischen Wahl oder Abstimmung auszählen. Mit dem *Auszählen* in der Tallier Applikation wird der Prozess gestartet. Nach Ende der Auszählung werden die Resultate in Form von Diagrammen angezeigt. Die folgende Auflistung zeigt die Hierarchie der Applikation anhand von Layers auf. Zu jedem Layer werden die dazugehörigen Pakete dargestellt, welche wiederum die Klassen enthalten, die das Auszählen einer Wahl oder Abstimmung und die Darstellung der jeweiligen Resultate ermöglichen:

- **UI-Layer**
  - gui
- **Application-Layer**
  - controller
- **Domain-Layer**
  - cryptomanager
  - connectionmanager
- **Service-Layer**
  - crypto
  - administration, collector, voting, authoroty, registration
  - configuration
  - commondata
  - datatypes
  - META-INF

4.3.1 Klassenübersicht

Das folgende Klassendiagrammm zeigt eine Übersicht aller Klassen und deren Abhängigkeiten auf. Die einzelnen Klassen werden in Anhang B.3 detailliert beschrieben und dargestellt. Die im Diagramm dargestellten hellblauen Pakete mit den violetten Klassen wurden im Rahmen dieses Projekts realisiert. Alle anderen Elemente, welche in einer anderen Farbe dargestellt werden, standen zur Verfügung und mussten nicht selbst implementiert werden.

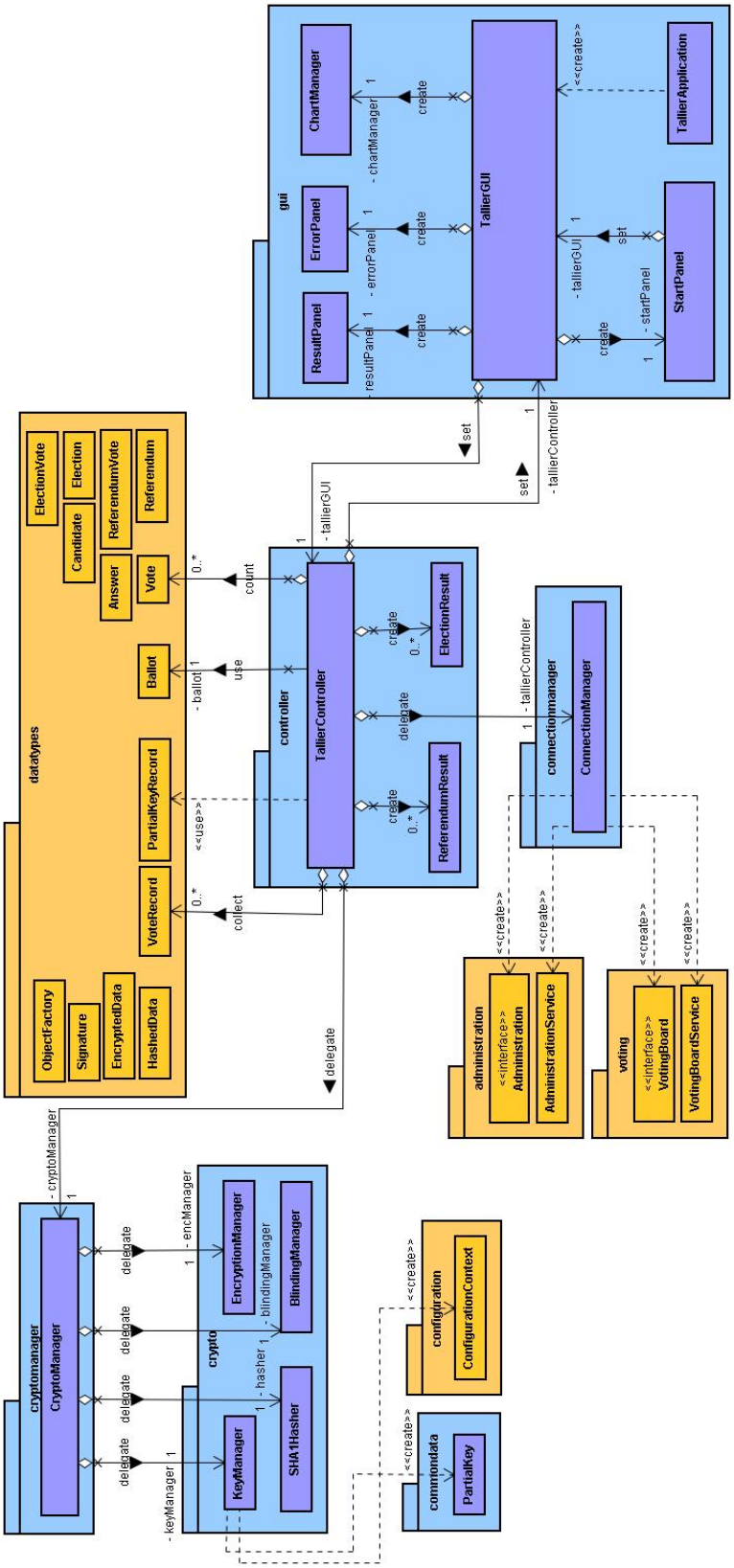

Abbildung 4.10: Übersicht Klassendiagramm des Talliers

4.3.2 Ablauf Stimmen auszählen

Das Auszählen der Stimmen gliedert sich in folgende Schritte:

1. Die zählende Person löst mit einem Klick auf den `Auszählen` Button den Auszählungsprozess aus.
2. Der aktuelle Wahlzettel wird über den ConnectionManager von dem Administration Service geholt.
3. Die `count` Methode wird im TallierController aufgerufen.
4. Die Public Keys der Authorities werden aus dem Java KeyStore extrahiert.
5. Die VoteRecords werden über den ConnectionManager vom Voting Board geholt.
6. Der Stimm-Kennwert und die verschlüsselte Stimme, die sich im VoteRecord befinden, werden zusammen mit der Event Nummber für jeden VoteRecord gehasht.
7. Jede Signatur des VoteRecords wird anhand des vorher entstandenen Hashwertes und dem Public Key der signierenden Authority verifiziert.
8. Aus den Teilschlüsseln jedes VoteRecords werden die geheimen Schlüssel rekonstruiert, die die Stimmen in der Voter Applikation verschlüsselt haben.
9. Jede Stimme aus dem VoteRecord wird entschlüsselt.
10. Alle Stimmen der Wahlen werden gezählt.
11. Alle Stimmen der Referenden werden gezählt.
12. Die Diagramme für sämtliche Wahlen und Referenden werden erstellt und im GUI dargestellt.

Die folgenden Abbildungen 4.11, 4.12 und 4.13 zeigen den detaillierten Sequenzablauf und alle beteiligten Klassen des Auszählungsprozesses.

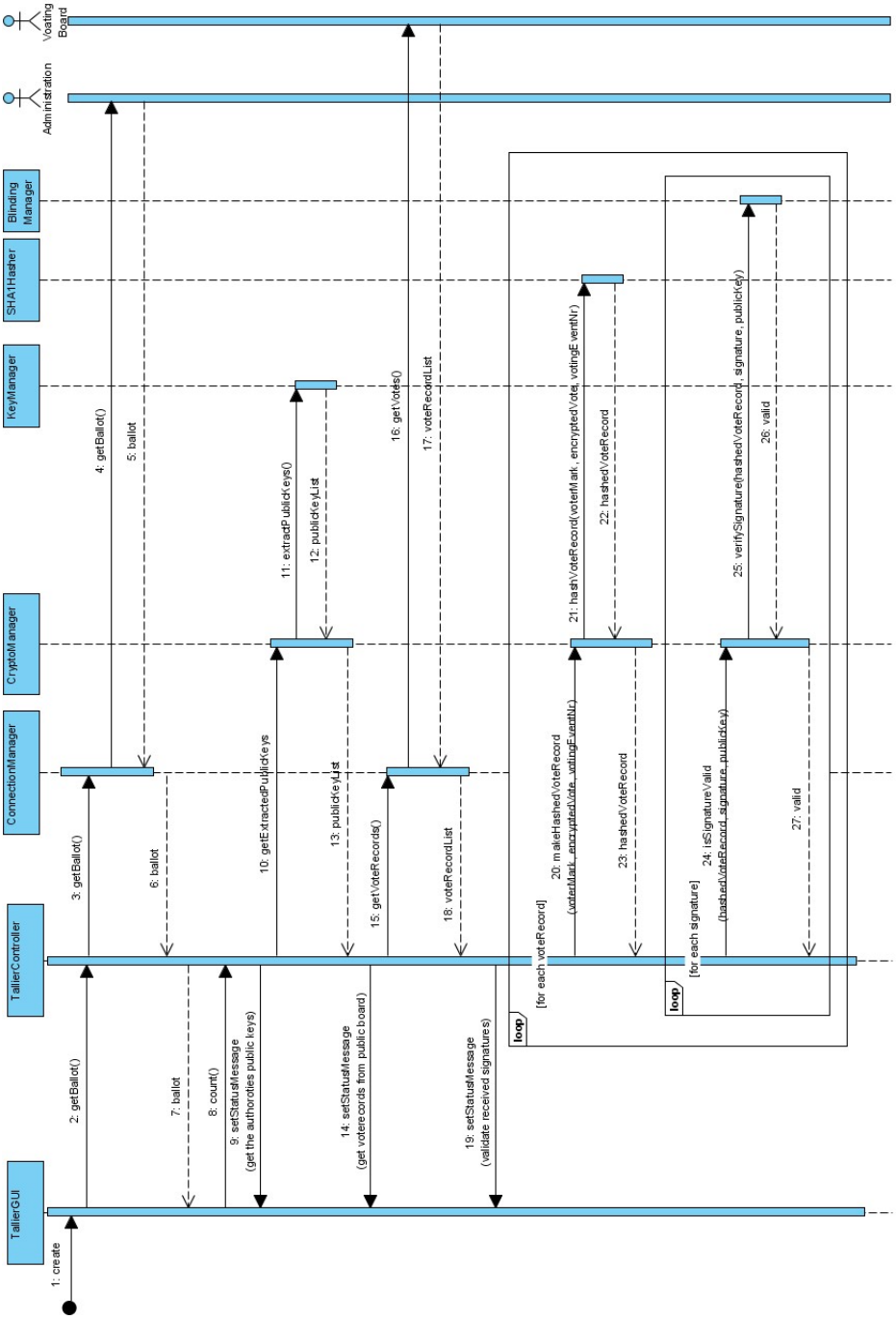

Abbildung 4.11: Sequenzdiagramm Teil 1 Stimmen auszählen

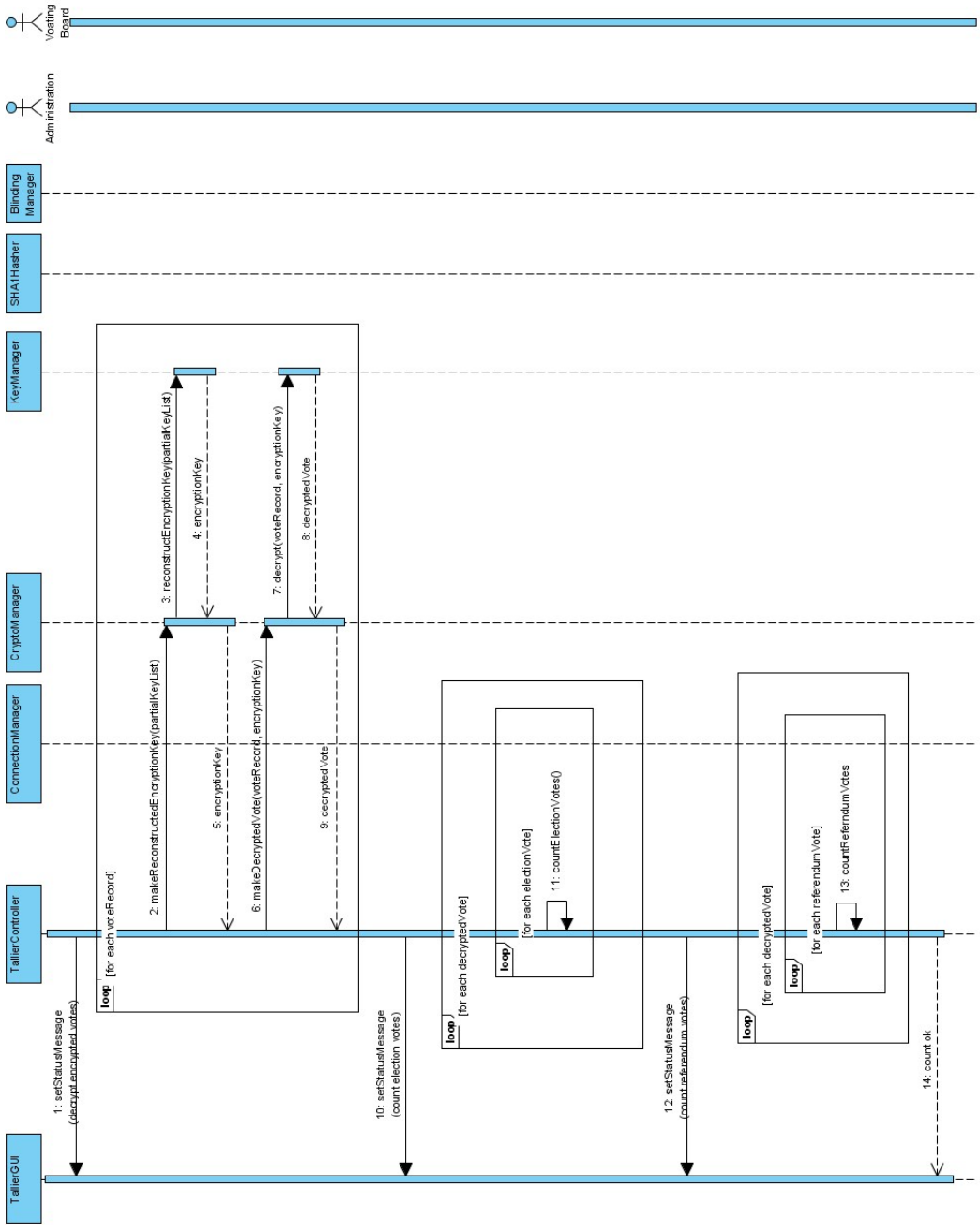

Abbildung 4.12: Sequenzdiagramm Teil 2 Stimmen auszählen

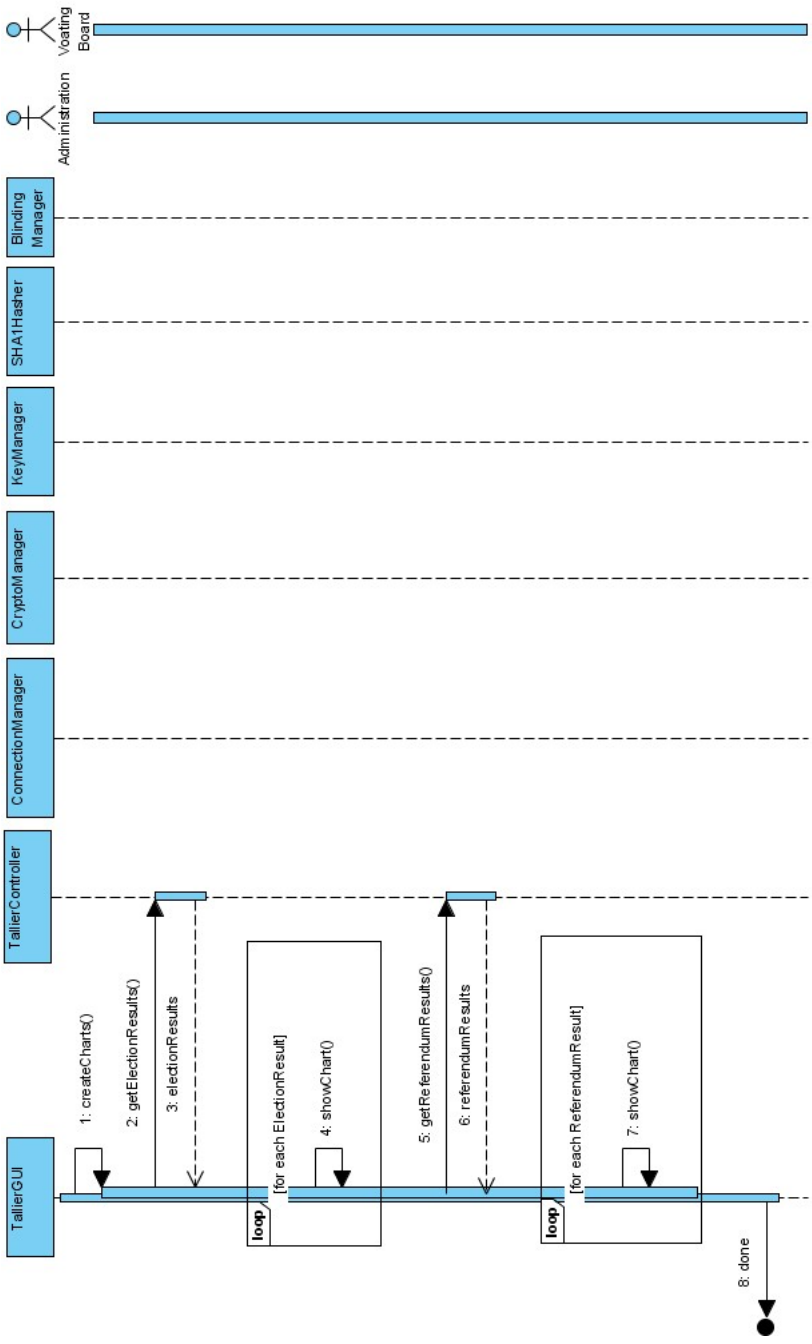

Abbildung 4.13: Sequenzdiagramm Teil 3 Stimmen auszählen

4.4 Exception Handling

Die Exceptions wurden grob in zwei Gruppen unterteilt. Diejenigen, die für einen Benutzer bzw. Administrator der Applikation interessant sein können und ihm auch angezeigt werden sollen und andere, die dem Benutzer zusammengefasst als eine *ApplicationFailureException* präsentiert werden. Folgende Exceptions werden dem Benutzer als *ApplicationFailureExeption* angezeigt:

- NullPointerException
- NoSuchAlgorithmException
- IOException
- JAXBException
- BadPaddingException
- IllegalBlockSizeException
- NoSuchPaddingException
- NoSuchProviderException
- MorePublicKeysThanBlindSignaturesException
- MoreBlindSignaturesThanPublicKeysException
- UnrecoverableKeyException
- CertificateException
- FileNotFoundException
- KeyStoreException
- MoreKeyCollectorsThanPartialKeysException
- MorePartialKeysThanCollectorsException

Folgende Exceptions informieren detaillierter über das Scheitern der Applikation:

- EventNotStartedException
- InsufficientPublicKeysException
- DuplicateVoterMarkException
- DuplicateVoterIdException
- SocketTimeoutException
- InsufficientValidSignaturesException
- InsufficientPartialKeysException
- InsufficientSignaturesException
- InvalidKeyException
- WebServiceException

Kapitel 5

Realisierung

5.1 Entwicklungsumgebung

Um die Wartung oder Weiterentwicklung des Systems zu erleichtern, wird im Folgenden die Entwicklungsumgebung näher beschrieben, welche zum Realisieren des Projekts benutzt wurde.

5.1.1 NetBeans

Das gesamte Projekt wurde mit NetBeans 6.5 entwickelt. Dabei wurde die Versionsverwaltung Subversion (SVN)[1] eingesetzt. Abbildung 5.1 zeigt die Projektansicht in NetBeans:

Folgende Projekte wurden im NetBeans Workspace angelegt:

- **Voter** Dieses Projekt beinhaltet die Source- und Ressource Dateien der Voter Applikation. Das Projekt besitzt Referenzen zum common-config-Projekt, um auf gemeinsam genutzte Konfiguration zugreifen zu können und auf das common-datatypes-Projekt. Die Voter Applikation besitzt eine ausführbare Klasse, `VoterApplication.java`, mit welcher die Applikation gestartet wird.

- **Tallier** Dieses Projekt beinhaltet die Source- und Ressource Dateien der Tallier Applikation. Das Projekt besitzt Referenzen zum common-config-Projekt, um auf gemeinsam genutzte Konfiguration zugreifen zu können und auf das common-datatypes-Projekt. Die Tallier Applikation besitzt eine ausführbare Klasse, `TallierApplication.java`, mit welcher die Applikation gestartet wird.

5.1.2 Zusätzliche Software

Im Rahmen des Projekts wurde folgende zusätzliche Software verwendet:

- JUDE Community 5.5 für das UML Design.

- Visual Paradigm for UML in den Versionen 6.4 und 7 für das UML-Design.

[1] Es wurde das BFH Subversion repository unter `https://svn.bfh.ch` verwendet

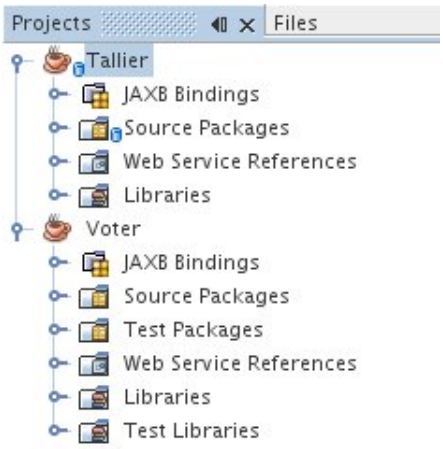

Abbildung 5.1: Übersicht NetBeans Projekte

- LaTeX[2] und die Entwicklungsumgebung Kile für das Erstellen von Artefakten.

5.2 Testhilfen

5.2.1 JUnit

Neben den manuellen Tests sind im Voter-Projekt automatisierte Tests mit JUnit integriert. Diese Tests sind darauf ausgerichtet, die Funktionalität der Service Klassen zu testen. Die JUnit-Tests sind nur im Voter-Projekt vorhanden, da die Voter vor der Tallier Applikation realisiert und in der Tallier Applikation die Service lassen der Voter Applikation übernommen wurden. Die Gesamttests des Systems wurden nicht mit JUnit, sondern manuell durchgeführt. Abbildung 5.2 zeigt eine Übersicht über die vorhandenen JUnit-Testklassen.

5.3 Eingesetzte Technologien

Die im Folgenden beschriebenen Technologien wurden für die Realisierung der Voter und Tallier Applikationen eingesetzt.

5.3.1 Advanced Encryption Standard

Der Advanced Encryption Standard (AES) ist ein symmetrisches Kryptosystem. AES oder auch Rijndael-Algorithmus, wurde in den 1990er Jahren von Joan Daemen und Vincent Rijmen entwickelt[5] und im Jahr 2000 vom US-amerikanischen National Institute of Standards and Technology (NIST) zum Nachfolger des Data Encryption Standard (DES) erklärt. Die Schlüssellänge kann auf 128, 192 oder 256 Bit festgelegt werden, wobei die Blocklänge immer 128 Bit beträgt. AES wird unter anderem vom Verschlüsselungsstandard IEEE 802.11i für Wireless LAN sowie bei Secure Shell

[2] Ein Softwarepaket, das die Benutzung des Textsatzprogramms TeX mit Hilfe von Makros vereinfacht, http://www.latex-project.org

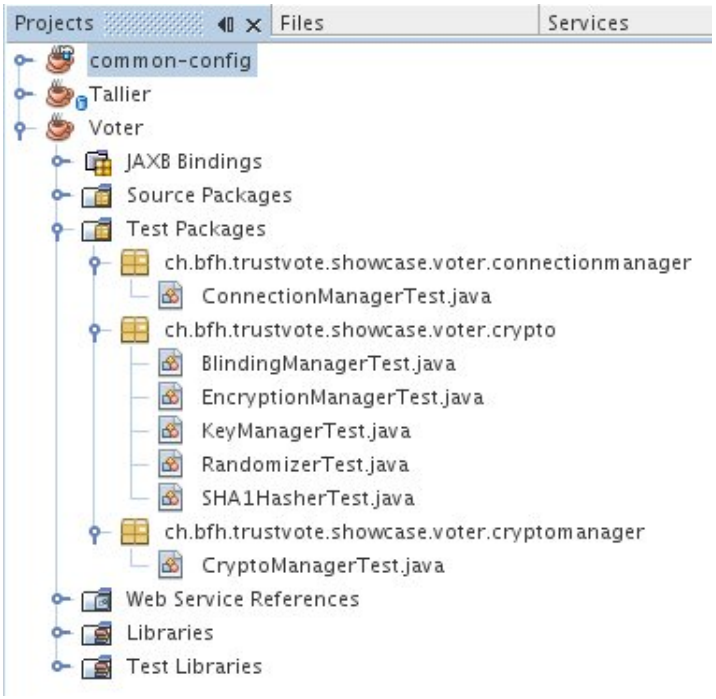

Abbildung 5.2: Übersicht JUnit-Tests

(SSH) und bei Internet Protocol Security (IPsec) genutzt. AES gilt bis heute als sicher und gehört zu den vom Projekt NESSIE des Information Society Technologies (IST) Programm der Europäischen Kommission empfohlenen kryptografischen Algorithmen[6].

5.3.2 Secure Hash Algorithm

Der Secure Hash Algorithm (SHA) ist ein kryptographischer Hash Algorithmus. SHA wurde in den 1990er Jahren vom NIST entwickelt[7]. Im Jahr 2005 wurde in SHA-1 Kollisionen gefunden[8]. NIST empfiehlt daher den Übergang von SHA-1 zu Hash-Funktionen der SHA-2-Familie (SHA-224, SHA-256, SHA-384, SHA-512)[9]. Für dieses Projekt wurde trotz der Empfehlung des NIST der Algorithmus SHA-1 verwendet, da dies in der `Common-Configuration` so definiert wurde.

5.3.3 RSA Algorithmus

In den 1970er Jahren wurde RSA von Ronald L. Rivest, Adi Shamir und Leonard Adleman am Massachusetts Institute of Technology in Cambridge (MIT) entwickelt[10], wobei RSA für die Anfangsbuchstaben der Namen der Autoren steh. RSA verwendet ein Schlüsselpaar bestehend aus einem privaten Schlüssel und einem öffentlichen Schlüssel. Der öffentliche Schlüssel wird zum Verschlüsseln und zum Verifizieren von Signaturen, der private Schlüssel zum Signieren und Entschlüsseln verwendet. RSA basiert darauf, dass die Faktorisierung einer grossen Zahl in ihre Primfaktoren sehr aufwändig ist, wogegen das Erzeugen einer Zahl durch Multiplikation zweier Primzahlen recht einfach ist. In der Voter wie auch in der Tallier Applikation wird RSA

dazu benutzt, die von den Authorities erhaltenen Signaturen zu verifizieren.

5.3.4 Shamir Secret Sharing

Shamir's Secret Sharing ist ein von Adi Shamir entwickeltes Secret-Sharing-Verfahren[4]. Mit Hilfe dieses Verfahrens ist es möglich, ein Geheimnis in mehrere Teilgeheimnisse aufzuteilen. Um das Geheimnis zu rekonstruieren, wird eine gewisse Anzahl Teilgeheimnisse benötigt.

5.3.5 (t, N)-Threshold Blind Signature Verfahren

In der Kryptographie bedeutet eine blinde Signatur eine Form der digitalen Signatur, bei welcher der Inhalt der zu signierenden Nachricht vorher verblindet wird. Die signierende Partei kennt somit den Inhalt der von ihr signierten Nachricht nicht. Das Konzept der blinden Signaturen wurde von David Chaum in den 1980er Jahren eingeführt[11]. Das in diesem Projekt verwendete (t, N)-Threshold Blind Signature Verfahren[3] lässt jede der `N` Parteien die blinden Daten signieren wobei mindestens `t` der resultierenden Signaturen gültig sein müssen, damit auch die blinden Daten als gültig angesehen werden.

5.3.6 Java Development Kit

Für alle Java-Komponenten wurde das JDK 1.6.0.13 zusammen mit den Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 6[3] eingesetzt.

Bei Verwendung des Java JDK müssen die JAR-Dateien *local_policy.jar* und *US_export_policy.jar* im Verzeichnis *<JAVA_HOME>/jre/lib/security* mit den heruntergeladenen Dateien ersetzt werden.

```
/opt/sun-jdk-1.6.0.13/jre/lib/security/
|-- US_export_policy.jar
|-- cacerts
|-- java.policy
|-- java.security
|-- javaws.policy
|-- local_policy.jar

0 directories, 6 files
```

5.3.7 Java Architecture for XML Binding

Java Architecture for XML Binding (JAXB) bezeichnet eine API in Java, die es ermöglicht, Daten aus einem XML-Schema automatisch an Java-Klassen zu binden.[12]

[3] Wegen amerikanischen Exportbeschränkungen ist die Verschlüsselung in Java auf 128 Bit beschränkt. Mit den Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 6 kann diese Exportbeschränkung aufgehoben werden. Die Dateien können unter `http://java.sun.com/javase/downloads/index.jsp` heruntergeladen werden.

Der JAXB Wizard von NetBeans wurde benutzt, um anhand der XML-Schemas der Datentypen, wie beispielsweise Ballot oder Vote, den Java Source-Code zu generieren

5.3.8 Bouncy Castle API

Bouncy Castle[4] ist ein freier Provider für die Java Cryptography Extension und die Java Cryptography Architecture. Bouncy Castle wurde aufgrund seiner Flexibilität genutzt, um die Ver- und Entschlüsselung von Stimmen zu implementieren. Aufgrund der einfacheren und somit weniger zeitintensiven Handhabung wurde nicht die Bouncy Castle Lightweight API, sondern der Bouncy Castle Security Provider eingesetzt.

5.3.9 JFreeChart

JFreeChart[5] ist ein Framework, mit welchem auf einfache Weise Diagramme erstellt werden können. Das Framework wurde benutzt, um mit der Tallier Applikation die Resultate einer Wahl oder Abstimmung in Form von Diagrammen darzustellen.

5.3.10 JCommon

JCommon[6] ist eine Java Library welche von JFreeChart benutzt wird. JCommon wird momentan nicht mehr aktiv weiterentwickelt und in Zukunft soll JFreeChart nicht mehr von dieser Library abhängig sein.

5.3.11 TableLayout

TableLayout[7] ist ein freier Layout Manager ähnlich dem bekannten GridBag Layout Manager. Wir haben TableLayout verwendet, um die Kandidierenden einer Wahl und deren jeweilige Beschreibung einfach darzustellen.

5.4 Implementations Details

Dieses Kapitel beschreibt einzelne Kernelemente der Voter und Tallier Applikationen. Das Schwergewicht liegt dabei auf der Realisierung, um einen Einblick in die Implementation zu geben.

5.4.1 (t, N)-Threshold Blind Signature Verfahren

Nach einigen Versuchen das ursprünglich im Protokoll vorgesehene (t, N)-Threshold Blind Signature Verfahren zu implementieren, wurden wir darauf aufmerksam, dass

[4] Mehr Informationen zu Bouncy Castle unter
http://www.bouncycastle.org
[5] Mehr Informationen zu JFreeChart unter
http://www.jfree.org/jfreechart
[6] Mehr Informationen zu JCommon unter
www.jfree.org/jcommon
[7] Mehr Informationen zu TableLayout unter
https://tablelayout.dev.java.net

dies aufgrund eines Fehlers in der Modulo-Rechnung beim Verblinden der Nachricht in dieser Form nicht möglich war. Das (t, N)-Threshold Blind Signature Verfahren musste umgestaltet werden und wurde, wie im folgenden Abschnitt beschrieben, in der Voter Applikation umgesetzt.
h ist die Nachricht, die verblindet werden soll und (e_i, N_i) mit $(i = 1 \dots N)$ die Public Keys der Authorities.

Zunächst wird die Nachricht h mit dem Blindierungsfaktor

$$r^{e_1 \cdots e_q} (\mathrm{mod} N_1 \cdots N_q) \tag{5.1}$$

unkenntlich gemacht, wobei r eine zufällig gewählte Zahl ist in der Grössenordnung von 128 Bit. Der Exponent und die anschliessende Modulorechnung angewendet auf r garantieren, dass jede Authority dieselben verblindeten Daten erhält. Das ist wichtig, damit alle blinden Signaturen nach der Entblindung mit dem individuellen Blindierungsfaktor einer Authority

$$r_i = r^{e_1 \cdots e_{i-1} e_{i+1} \cdots e_q} (\mathrm{mod} N_i) \tag{5.2}$$

gültig sind.
Die Gleichung 5.3 zeigt die Kalkulation woraus die blinden Daten h' resultieren.

$$h' = h \cdot r^{e_1 \cdots e_q} (\mathrm{mod} N_1 \cdots N_q) \tag{5.3}$$

Listing 5.1 aus der Klasse `BlindingManager` beinhaltet die Verblindung einer Nachricht (Zeile 1) mit den zugehörigen Hilfsmethoden `multiplyPKModulus`, die sämtliche Modulos der Public Keys miteinander mulitpliziert und `multiplyPKExponent`, die das Gleiche mit den Exponent macht (Zeilen 14 und 29). Damit die Berechnungen auf die Nachricht h angewendet werden können, muss diese zuerst in einen BigInteger umgewandelt werden. Dabei muss ein positiver BigInteger entsteht, da bei einem negativen Wert sämtliche entblindeten Signaturen ungültig sind. Dies geschieht durch das Setzen des Sign-Bits im BigInteger-Konstruktor (Zeile 6). Das Sign-Bit kann mit -1 (für negative Zahlen), 0 (Null) und 1 (für positive Zahlen) belegt werden. Dies muss ebenfalls auf den Web Services beim Signieren der verblindeten Nachricht beachtet werden.

```
1  public byte[] blindMessage(BigInteger blindingFactor, HashedData
       hashedVoteRecord, HashMap<String, RSAPublicKey> publicKeyList)
2    throws InsufficientPublicKeysException {
3    if (publicKeyList.size() == NUMBER_OF_AUTHOROTIES) {
4      BigInteger pkModulusProduct = multiplyPKModulus(publicKeyList);
5      BigInteger pkExponentProduct = multiplyPKExponent(publicKeyList);
6      BigInteger blindedMessage = new BigInteger(1, hashedVoteRecord.getValue())
7        .multiply(blindingFactor.modPow(pkExponentProduct, pkModulusProduct)).
             mod(pkModulusProduct);
8      return blindedMessage.toByteArray();
9    } else {
10     throw new InsufficientPublicKeysException("Insufficient public keys!");
11   }
12 }
```

```
13
14  private BigInteger multiplyPKModulus(HashMap<String, RSAPublicKey> pkList) {
15    Iterator iter = pkList.keySet().iterator();
16    String index = (String) iter.next();
17    RSAPublicKey firstKey = pkList.get(index);
18    BigInteger result = firstKey.getModulus();
19    for (RSAPublicKey pk : pkList.values()) {
20      RSAPublicKey tmpKey = pk;
21      if (!tmpKey.equals(firstKey)) {
22        BigInteger tmpResult = result.multiply(tmpKey.getModulus());
23        result = tmpResult;
24      }
25    }
26    return result;
27  }
28
29  private BigInteger multiplyPKExponent(HashMap<String, RSAPublicKey> pkList) {
30    Iterator iter = pkList.keySet().iterator();
31    String index = (String) iter.next();
32    RSAPublicKey firstKey = pkList.get(index);
33    BigInteger result = firstKey.getPublicExponent();
34    for (RSAPublicKey pk : pkList.values()) {
35      RSAPublicKey tmpKey = pk;
36      if (!tmpKey.equals(firstKey)) {
37        result = result.multiply(tmpKey.getPublicExponent());
38      }
39    }
40    return result;
41  }
```

Listing 5.1: Verblinden einer Nachricht

Listing 5.2 aus der Klasse `BlindingManager` zeigt die Methode `createPartialBlindingFactor`, die die individuellen Blindierungsfaktoren anhand der Gleichung 5.2 berechnet.

```
1  public HashMap<String, BigInteger> createPartialBlindingFactor(BigInteger
       blindingFactor, HashMap<String, RSAPublicKey> publicKeyList)
2    throws InsufficientPublicKeysException {
3    if (publicKeyList.size() == NUMBER_OF_AUTHOROTIES) {
4      HashMap<String, BigInteger> partialBlindingFactorList = new HashMap<String
         , BigInteger>();
5      for (String index : publicKeyList.keySet()) {
6        HashMap<String, RSAPublicKey> tmpPKList = new HashMap<String,
           RSAPublicKey>(publicKeyList);
7        RSAPublicKey removedPK = tmpPKList.remove(index);
8        BigInteger pkExponentProduct = multiplyPKExponent(tmpPKList);
9        partialBlindingFactorList.put(index, blindingFactor.modPow(
           pkExponentProduct, removedPK.getModulus()));
10     }
11     return partialBlindingFactorList;
12   } else {
13     throw new InsufficientPublicKeysException("Insufficient public keys!");
14   }
15 }
```

Listing 5.2: Berechnung der Blindierungsfaktoren

Die von den Authorities erzeugte blinde Signatur wird gemäss Gleichung 5.4 entblindet.

$$s_i = \text{unblind}(s'_i, r_i) = s'_i \cdot r_i^{-1} (\text{mod} N_i) \equiv h^{d_i} (\text{mod} N_i) = \text{sign}_{d_i}(h) \qquad (5.4)$$

Listing 5.3 aus der Klasse `BlindingManager` zeigt die Methode `unblindSignature`, die sämtliche blinden Signaturen mit den Blindierungsfaktoren der Authorities entblindet.

```
1  public HashMap<String, Signature> unblindSignature(HashMap<String, Signature>
       blindSignatureList,
2    HashMap<String, BigInteger> blindingFactorList, HashMap<String, RSAPublicKey
       > publicKeyList)
3    throws MoreBlindSignaturesThanPublicKeysException,
       MorePublicKeysThanBlindSignaturesException {
4    if (blindSignatureList.size() == publicKeyList.size()) {
5      ObjectFactory objFactory = new ObjectFactory();
6      HashMap<String, Signature> unblindSignatureList = new HashMap<String,
         Signature>();
7      for (String index : blindSignatureList.keySet()) {
8        Signature unblindSignature = objFactory.createSignature();
9        BigInteger unblindedSignedMessageAuth = new BigInteger(
           blindSignatureList.get(index).getValue()).multiply(
           blindingFactorList.get(index).modInverse(publicKeyList.get(index).
           getModulus())).mod(publicKeyList.get(index).getModulus());
10       unblindSignature.setValue(unblindedSignedMessageAuth.toByteArray());
11       unblindSignature.setAlgorithm(blindSignatureList.get(index).getAlgorithm
           ());
12       unblindSignature.setSignerId(blindSignatureList.get(index).getSignerId()
           );
13       unblindSignatureList.put(index, unblindSignature);
14     }
15     return unblindSignatureList;
16   } else if (blindSignatureList.size() > publicKeyList.size()) {
17     throw new MoreBlindSignaturesThanPublicKeysException("There are more blind
         signatures available than public keys!");
18   } else {
19     throw new MorePublicKeysThanBlindSignaturesException("There are more
         public keys available than blind signatures!");
20   }
21 }
```

Listing 5.3: Entblinden einer Nachricht

5.4.2 Shamir Secret Sharing

Um einen geheimen Schlüssel in mehrere Teilschlüssel zu unterteilen wird zuerst ein Polynom vom Grad $k-1$ gewählt. k bezeichnet die Anzahl nötige Teilschlüssel, um den geheimen Schlüssel zu rekonstruieren. Das Polynom hat dabei die Form

$$f(x) = s + a_1 \cdot x + a_2 \cdot x^2 + \ldots + a_{k-1} \cdot x^{k-1} \tag{5.5}$$

wobei s der geheime Schlüssel ist und die Koeffizienten a_i zufällig gewählt werden. Der geheime Schlüssel kann nun in n Teilschlüssel aufgeteilt werden, indem jedes n auf $f(x)$ angewendet wird. So werden n Punkte $(x, f(x))$ generiert. Damit diese Punkte zum Wiederherstellen des geheimen Schlüssels verwendet werden können, werden diese in ein serialisierbares PartialKey-Objekt abgelegt. Dieses Objekt ist serialisiert, da es in ein Byte-Array umgewandelt werden muss, weil der PartielKeyRecord, der an die Collectors versendet wird, nur ein Byte-Array als PartialKey akzeptiert.

Listing 5.4 aus der Klasse `KeyManager` zeigt das Aufteilen des geheimen Schlüssels

mit der `createPartialKeys` Methode (Zeile 1). Die verwendet die Hilfsmethode `convertToByteArray`, um aus einem PartialKey-Objekt ein Byte-Array zu erzeugen (Zeile 30) und die Hilfsmethode`poly` (Zeile 37) die das Resultat des eingesetzten Wertes in die Polynomialfunktion zurück gibt.

```
1  public List<byte[]> createPartialKeys(SecretKey key) throws IOException {
2    ArrayList<byte[]> partialKeyList = new ArrayList<byte[]>();
3    SecureRandom random = new SecureRandom();
4    /* Number of bits to create the prime number.
5    * The prime number must be greater than the diveded secret */
6    int numBits = key.getEncoded().length * 8 + 1;
7    this.primeNumber = BigInteger.probablePrime(numBits, random);
8    BigInteger secret = new BigInteger(key.getEncoded());
9    /* coefficients (a0,...,at-1) are part of the polynomial function.
10   For example: f(x) = a0 + a1*x + a2*x^2 */
11   BigInteger[] coefs = new BigInteger[PARTIALKEY_THRESHOLD];
12   /* the first coefficient is always the sharing secret */
13   coefs[0] = secret;
14   /* the coefficients will be created */
15   for(int j=0; j<PARTIALKEY_THRESHOLD; j++) {
16     if(j!=0){
17     coefs[j] = new BigInteger(numBits-1,random);
18     }
19   }
20   /* the partial keys for n parties will be created */
21   for(int i=0; i < NUMBER_OF_COLLECTORS; i++) {
22     SecretKey shareKey = new SecretKeySpec(poly(coefs,i+1).toByteArray(), "AES
         ");
23     PartialKey partialKey = new PartialKey(shareKey, i, this.primeNumber);
24     byte[] partialKeyInByte = KeyManager.convertToByteArray(partialKey);
25     partialKeyList.add(i, partialKeyInByte);
26   }
27   return partialKeyList;
28 }
29
30 public static byte[] convertToByteArray(PartialKey pk) throws IOException {
31   ByteArrayOutputStream baos = new ByteArrayOutputStream();
32   ObjectOutputStream oos = new ObjectOutputStream(baos);
33   oos.writeObject(pk);
34   return baos.toByteArray();
35 }
36
37 private BigInteger poly(BigInteger[] coefs, int val) {
38   BigInteger y = coefs[0];
39   for(int j=1; j<coefs.length; j++){
40     BigInteger term = coefs[j].multiply(BigInteger.valueOf((long)Math.pow(val,
         j)));
41     y = y.add(term);
42   }
43   return y.mod(this.primeNumber);
44 }
```

Listing 5.4: Schlüssel aufteilen

Mit Hilfe der Lagrange[8] Polynome

$$l_i(x) = \prod_{j=0, j\neq i}^{n} \frac{(x - x_j)}{(x_i - x_j)} \tag{5.6}$$

[8] Joseph-Louis de Lagrange (1736-1813): italienischer Mathematiker und Astronom

kann nun mit Funktion 5.7, vorausgesetzt es sind genügend Teilschlüssel vorhanden, der geheime Schlüssel wiederhergestellt werden.

$$f(x) = \sum_{y=0}^{2} y_j \cdot l_j(x) \tag{5.7}$$

Listing 5.5 aus der Klasse `KeyManager` zeigt das Wiederherstellen des geheimen Schlüssels (Zeile 1), wobei die Berechnung des Lagrange Polynoms auf Zeile 14 passiert, mit der entsprechenden Hilfsmethode (Zeile 27).

```
1  public SecretKey reconstructEncryptionKey(List<byte[]> partialKeyList) throws
       InsufficientPartialKeysException,
2    IOException, ClassNotFoundException {
3    if (partialKeyList.size() >= PARTIALKEY_THRESHOLD) {
4      Map<Integer, PartialKey> sortedPartialKeyList = new TreeMap<Integer,
         PartialKey>();
5      for (int i = 0; i < partialKeyList.size(); i++) {
6        PartialKey tmpPK = KeyManager.convertToPartialKey(partialKeyList.get(i))
           ;
7        sortedPartialKeyList.put(tmpPK.getXValue(), tmpPK);
8      }
9      BigInteger secret = BigInteger.ZERO;
10     for (Integer index : sortedPartialKeyList.keySet()) {
11       float lambda = 1;
12       for (Integer index2 : sortedPartialKeyList.keySet()) {
13         if (index2 != index) {
14           lambda = lambda * ((float) (index2 + 1) / (float) (index2 - index));
15         }
16       }
17       PartialKey partialKey = sortedPartialKeyList.get(index);
18       BigInteger partialKeyValue = new BigInteger(partialKey.getPartialKey().
           getEncoded());
19       secret = secret.add(partialKeyValue.multiply(BigInteger.valueOf((long)
           lambda))).mod(partialKey.getPrimeNumber());
20     }
21   return new SecretKeySpec(secret.toByteArray(), "AES");
22   } else {
23     throw new InsufficientPartialKeysException("Not enough valid partial keys
         to reconstruct the encryption key!");
24   }
25 }
26
27 public static PartialKey convertToPartialKey(byte[] b) throws IOException,
     ClassNotFoundException {
28   ByteArrayInputStream bais = new ByteArrayInputStream(b);
29   ObjectInputStream ois = new ObjectInputStream(bais);
30   return (PartialKey)ois.readObject();
31 }
```

Listing 5.5: Schlüssel wiederherstellen

5.4.3 Web Service Referenzen

Die benötigten Referenzen zu den Web Services wurden alle mit NetBeans aus den bereits vorhandenen WSDL-Files der Web Services generiert. Abbildung 5.3 zeigt die generierten Referenzen.

Die Verbindungen zu den Web Services werden in der Klasse `ConnectionManager`

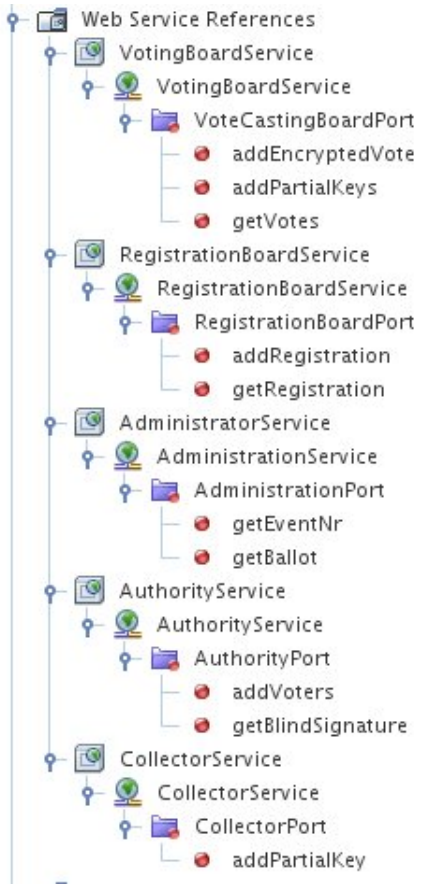

Abbildung 5.3: Übersicht Web Service References

hergestellt. Listing 5.6 zeigt einen Auszug der Methoden dieser Klasse. Um das Erstellen von mehreren identischen Web Service Referenzen zu vermeiden und eine grössere Flexibilität im Umgang mit den URLs der Web Services zu gewährleisten, wird bei Methoden, welche mehrere verschiedene Web Services ansprechen, wie beispielsweise `sendPartialKeyRecord` (Zeile 1), die URL des Web Services übergeben. Diese wird anschliessend vom BindingProvider zum Aufbau der Verbindung genutzt (Zeile 7). Die URLs zu den Web Services sind in einer XML-Datei im Projekt common-config definiert, aus welchem die Klasse ConfigurationContext generiert wurde. Nach Instanziieren eines ConfigurationContext-Objekt können die URLs aus diesem gelesen und den entsprechenden Methoden übergeben werden.

```
1  public void sendPartialKeyRecord(PartialKeyRecord partialKeyRecord, String url
       ) throws
2    DuplicateVoterMarkException, SocketTimeoutException, WebServiceException {
3    ch.bfh.trustvote.showcase.collector.CollectorService service = new ch.bfh.
         trustvote.showcase.collector.CollectorService();
4    ch.bfh.trustvote.showcase.collector.Collector port = service.
         getCollectorPort();
5    BindingProvider bp = (BindingProvider) port;
6    bp.getRequestContext().put("com.sun.xml.ws.request.timeout", new Integer
         (10000));
7    bp.getRequestContext().put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY, url);
8    port.addPartialKey(partialKeyRecord);
9  }
10
```

```
11 public void sendEncryptedVoteRecord(EncryptedVoteRecord encryptedVoteRecord)
       throws
12   ch.bfh.trustvote.showcase.voting.DuplicateVoterMarkException,
13   SocketTimeoutException, WebServiceException {
14   ch.bfh.trustvote.showcase.voting.VotingBoardService service = new ch.bfh.
         trustvote.showcase.voting.VotingBoardService();
15   ch.bfh.trustvote.showcase.voting.VotingBoard port = service.
         getVoteCastingBoardPort();
16   BindingProvider bp = (BindingProvider) port;
17   bp.getRequestContext().put("com.sun.xml.ws.request.timeout", new Integer
         (10000));
18   port.addEncryptedVote(encryptedVoteRecord);
19 }
```

Listing 5.6: Auszug ConnectionManager

5.4.4 Ver- und Entschlüsseln

Aus zeitlichen Gründen wurde auf die Implementation von verschiedenen Algorithmen zum Verschlüsseln der Stimmen verzichtet. Da die Stimmen von der Voter Applikation ver- und von der Tallier Applikation entschlüsselt werden, kann sichergestellt werden, dass keine ungewünschten Algorithmen zur Verschlüsselung der Stimmen benutzt werden.

Um Stimmen zu verschlüsseln haben wir uns auf die Verwendung von AES mit der Schlüssellänge 256 Bit festgelegt. Als Betriebsart wurde Counter Mode (CTR)[9] und als Padding-Methode PKCS7[10] gewählt.

Listing 5.7 zeigt einen Auszug aus der Klasse `EncryptionManager` mit der Methode zum Verschlüsseln einer Stimme (Zeile 1) sowie einer Hilfsmethode, welche das Stimm-Objekte in XML-Daten serialisiert (Zeile 19).

```
1  public EncryptedData encrypt(Vote vote, Key key) throws
       NoSuchAlgorithmException,
2    NoSuchPaddingException, InvalidKeyException, IllegalBlockSizeException,
3    BadPaddingException, JAXBException, NoSuchProviderException {
4    //Marshal vote
5    byte[] data = marshal(vote);
6    //Init cipher for encryption
7    Cipher cipher = Cipher.getInstance("AES/CTR/PKCS7Padding", "BC");
8    cipher.init(Cipher.ENCRYPT_MODE, key);
9    //Encrypte marshalled vote
10   byte[] encryptedData = cipher.doFinal(data);
11   //Create encryptedData Object
12   EncryptedData encData = new EncryptedData();
13   encData.setAlgorithm(EncryptionAlgorithm.AES_256);
14   encData.setInitVector(cipher.getIV());
15   encData.setValue(encryptedData);
16   return encData;
17 }
18
19 private static byte[] marshal(Vote vote) throws JAXBException {
20   String pkgname = Vote.class.getPackage().getName();
21   JAXBContext context = JAXBContext.newInstance(pkgname);
```

[9] Counter Mode (CTR) ist eine Betriebsart, in der Blockchiffren betrieben werden können[13].

[10] Bei symmetrischen und asymmetrischen Blockchiffren wird Padding verwendet, um den Klartext an die Blocklänge anzupassen. PKCS7 Padding[15] ist eine Erweiterung zu PKCS5 Padding[14], welche auch bei Blockchiffren mit einer Blockgrösse von mehr als acht Byte angewendet werden kann

```
22    Marshaller marshaller = context.createMarshaller();
23    marshaller.setProperty(Marshaller.JAXB_FRAGMENT, true);
24    QName qname = new QName("http://showcase.trustvote.bfh.ch/datatypes/", "vote
          ");
25    JAXBElement<Vote> element = new JAXBElement<Vote>(
26      qname, Vote.class, vote);
27    ByteArrayOutputStream bos = new ByteArrayOutputStream();
28    marshaller.marshal(element, new StreamResult(bos));
29    return bos.toByteArray();
30  }
```

Listing 5.7: Verschlüsseln einer Stimme

Listing 5.8 zeigt einen Auszug aus der Klasse `EncryptionManager` mit der Methode zum Entschlüsseln einer Stimme (Zeile 1) wobei die XML-Daten nach dem Entschlüsseln in ein Stimm-Objekt serialisiert werden (Zeile 17).

```
1   public Vote decrypt(VoteRecord voteRecord, Key reconstructedKey) throws
        NoSuchAlgorithmException, NoSuchPaddingException,
2     InvalidKeyException, InvalidKeyException, InvalidAlgorithmParameterException
          ,
3     IllegalBlockSizeException, BadPaddingException, JAXBException,
          NoSuchProviderException {
4     //Get encrypted data
5     EncryptedData encryptedVote = voteRecord.getEncryptedVote();
6     byte[] encryptedData = encryptedVote.getValue();
7     IvParameterSpec ivSpec = new IvParameterSpec(encryptedVote.getInitVector());
8     //Init cipher
9     Cipher cipher = Cipher.getInstance("AES/CTR/PKCS7Padding", "BC");
10    cipher.init(Cipher.DECRYPT_MODE, reconstructedKey, ivSpec);
11    //Decrypt vote
12    byte[] decryptedData = cipher.doFinal(encryptedData);
13    //Unmarshal and return vote
14    return unmarshal(decryptedData);
15  }
16
17  private static Vote unmarshal(byte[] data) throws JAXBException {
18    String pkgname = Vote.class.getPackage().getName();
19    JAXBContext context = JAXBContext.newInstance(pkgname);
20    Unmarshaller unmarshaller = context.createUnmarshaller();
21    ByteArrayInputStream bis = new ByteArrayInputStream(data);
22    JAXBElement<Vote> element = (JAXBElement<Vote>)unmarshaller.unmarshal(new
          StreamSource(bis), Vote.class);
23    return element.getValue();
24  }
```

Listing 5.8: Entschlüsseln einer Stimme

Kapitel 6

Tests

6.1 Einleitung

Hauptziel der Tests ist die Überprüfung der Funktionalitäten und damit die Erbringung des Nachweises, dass diese gemäss den Anforderungen implementiert wurden. Weiter dienen die Tests der Qualitätssicherung. Sie ermöglichen das Auffinden von Fehlern in der Analyse, dem Design sowie in der Realisierung. Fehlfunktionen in den Voter und Tallier Applikationen sollen verhindert werden. Der Testaufbau sowie die definierten Testfälle werden in den nachfolgenden Kapiteln erläutert.

6.2 Testumgebung

Die Testumgebung beschreibt die Hardware, welche zur Durchführung der Tests eingesetzt wurde. Die Voter und Tallier Applikationen waren jeweils beide auf dem selben Computer installiert. Die für die Durchführung einer Abstimmung oder Wahl nötigen Web Services sind auf virtuellen Maschinen im Netzwerk der BFH installiert. Auf die Web Services kann von ausserhalb der BFH zugegriffen werden. Abbildung 6.1 zeigt den Aufbau der Testumgebung.

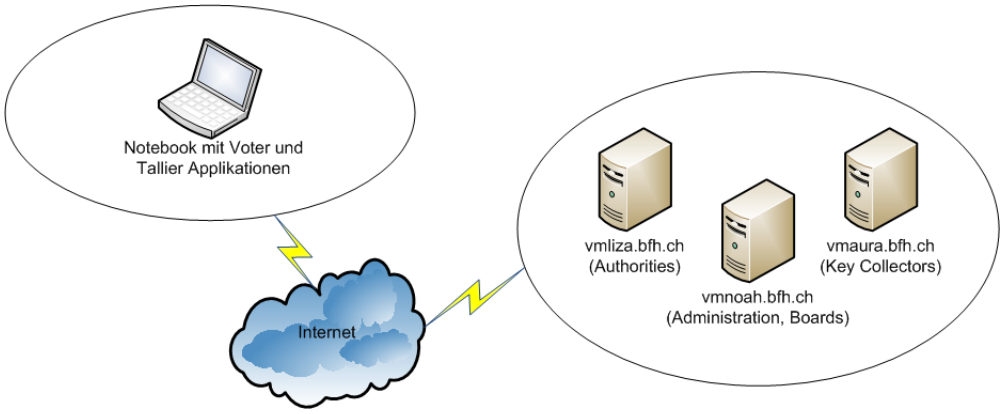

Abbildung 6.1: Überblick Testumgebung

6.3 Testaufbau

Der Testaufbau beschreibt erstens, welche Testarten durchgeführt wurden und zweitens, welche Bereiche des Systems getestet wurden. Wir führten ausschliesslich Funktionstests durch. Mit den Funktionstests werden die funktionalen Anforderungen an das System gemäss den Use Cases überprüft. Im Rahmen der Funktionstests testeten wir jeweils die Voter und Tallier Applikation.

6.4 Vorgehen

In einem ersten Schritt werden während der Entwicklung die einzelnen Service-Klassen, wie beispielsweise die Klasse `EncryptionManager`, des Voters getestet. Da die Tallier Applikation auf die selben Service-Klassen zurückgreift, werden nur diejenigen der Voter Applikation getestet. Diese Einzeltests werden automatisiert mit JUnit durchgeführt und beim Erstellen der jeweiligen Klasse definiert. Sie sind hier nicht weiter aufgeführt. In einem zweiten Schritt werden die Funktionstests in Bezug auf die Voter und Tallier Applikationen durchgeführt. Abbildung 6.2 gibt einen Überblick über den Testaufbau wie auch über das Testvorgehen.

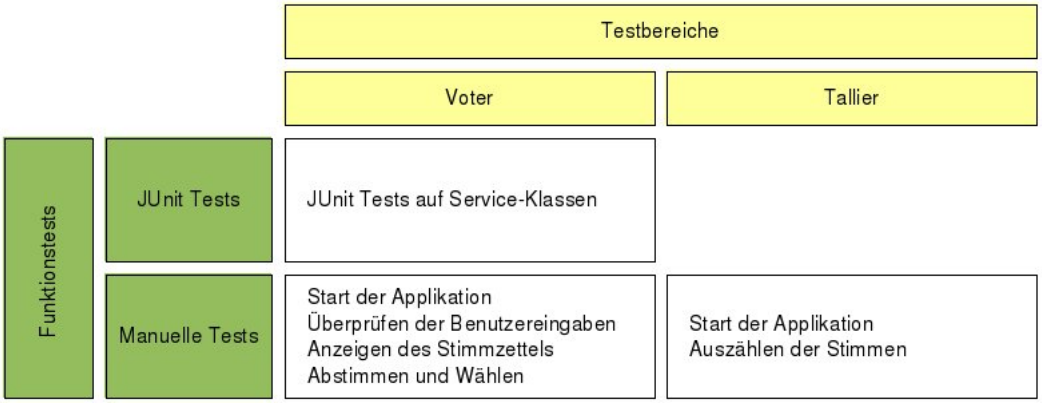

Abbildung 6.2: Überblick Testaufbau und Testvorgehen

Die Tests haben am Donnerstag 4. Juni 2009 stattgefunden. Die Resultate der Tests sind in den Testfällen in Abschnitt 6.5 aufgelistet.

6.5 Funktionstests

6.5.1 Voter

Start der Applikation

Variante	Erwartetes Resultat	Tatsächliches Resultat	Befund
Internetverbindung vorhanden	Applikation startet nach dem Abholen des Stimmzettels bei der Administration	Applikation startet nach dem Abholen des Stimmzettels bei der Administration	OK
Keine Internetverbindung vorhanden	Applikation startet, Fehlermeldung wird angezeigt	Applikation startet, Fehlermeldung wird angezeigt	OK
Der Stimmzettel enthält keine Abstimmungen oder Wahlen	Fehlermeldung beim Start der Applikation wird angezeigt	Fehlermeldung beim Start der Applikation wird angezeigt	OK
Voting Event hat noch nicht begonnen	Fehlermeldung beim Start der Applikation wird angezeigt	Fehlermeldung beim Start der Applikation wird angezeigt	OK

Tabelle 6.1: Start der Applikation

Überprüfen der Benutzerangaben

Variante	Erwartetes Resultat	Tatsächliches Resultat	Befund
Einloggen mit korrekten Benutzerangaben	Login erfolgreich, Wechsel zur ersten Abstimmung oder Wahl	Login ist erfolgreich, Wechsel zur ersten Abstimmung oder Wahl	OK
Einloggen mit einem falschen Benutzernamen	Login nicht erfolgreich, Hinweis wird angezeigt	Login nicht erfolgreich, Hinweis wird angezeigt	OK
Einloggen mit einem falschen Passwort	Login nicht erfolgreich, Hinweis wird angezeigt	Login nicht erfolgreich, Hinweis wird angezeigt	OK
Einloggen mit einer falscher Prüfsumme	Login nicht erfolgreich, Hinweis wird angezeigt	Login nicht erfolgreich, Hinweis wird angezeigt	OK

Tabelle 6.2: Überprüfen der Benutzerangaben

Anzeigen des Stimmzettels

Variante	Erwartetes Resultat	Tatsächliches Resultat	Befund
Stimmzettel mit einer Abstimmung	Abstimmung wird angezeigt, Navigation funktioniert einwandfrei	Abstimmung wird angezeigt, Navigation funktioniert einwandfrei	OK
Stimmzettel mit einer Abstimmung und einer Wahl	Abstimmung und Wahl werden angezeigt, Navigation funktioniert einwandfrei	Abstimmung und Wahl werden angezeigt, Navigation funktioniert einwandfrei	OK
Stimmzettel mit einer Abstimmung und zwei Wahlen	Abstimmung und Wahlen werden angezeigt, Navigation funktioniert einwandfrei	Abstimmung und Wahlen werden angezeigt, Navigation funktioniert einwandfrei	OK

Tabelle 6.3: Anzeige des Stimmzettels

Abstimmen und wählen

Variante	Erwartetes Resultat	Tatsächliches Resultat	Befund
Kein Kandidierender muss gewählt werden, es kann maximal ein Kandidierender gewählt werden	Auswahl mit Radio Buttons	Auswahl mit Radio Buttons	OK
Kein Kandidierender muss gewählt werden, es können maximal fünf Kandidierende gewählt werden	Auswahl mit Checkboxes, wenn maximale Anzahl erreicht, können keine weiteren gewählt werden	Auswahl mit Checkboxes, wenn maximale Anzahl erreicht, werden die verbleibenden Kandidierenden deaktiviert	OK
Minimum ein Kandidierender muss gewählt werden, es können maximal fünf Kandidierende gewählt werden	Auswahl mit Checkboxes, solange kein Kandidierender gewählt kann nicht fortgefahren werden. Wenn maximale Anzahl erreicht, können keine weiteren gewählt werden	Auswahl mit Checkboxes, solange kein Kandidierender gewählt ist “Weiter” deaktiviert. Wenn maximale Anzahl erreicht, werden die verbleibenden Kandidierenden deaktiviert	OK
Java Cryptography Extension Unlimited Strength Jurisdiction Policy Files sind installiert	Schlüssel können generiert und Stimme verschlüsselt werden	Schlüssel werden generiert und Stimme verschlüsselt	OK
Eine Stimme wird erfolgreich abgegeben	Stimme wird abgegeben und auf dem Public Board publiziert. Teilschlüssel sind bei Collectors vorhanden	Stimme wird abgegeben und auf dem Public Board publiziert. Die Teilschlüssel sind bei den Collectors vorhanden	OK

Tabelle 6.4: Abstimmen und wählen

6.5.2 Tallier

Start der Applikation

Variante	Erwartetes Resultat	Tatsächliches Resultat	Befund
Internetverbindung vorhanden	Applikation startet nach dem Abholen des Stimmzettels bei der Administration	Applikation startet nach dem Abholen des Stimmzettels bei der Administration	OK
Keine Internetverbindung vorhanden	Applikation startet, Fehlermeldung wird angezeigt	Applikation startet, Fehlermeldung wird angezeigt	OK
Der Stimmzettel enthält keine Abstimmung oder Wahl	Fehlermeldung beim Start der Applikation wird angezeigt	Fehlermeldung beim Start der Applikation wird angezeigt	OK
Voting Event hat noch nicht begonnen	Fehlermeldung beim Start der Applikation wird angezeigt	Fehlermeldung beim Start der Applikation wird angezeigt	OK

Tabelle 6.5: Start der Applikation

Auszählen der Stimmen

Variante	Erwartetes Resultat	Tatsächliches Resultat	Befund
Eine getätigte Abstimmung wird ausgezählt	Stimmen können entschlüsselt und gezählt werden	Stimmen werden entschlüsselt und gezählt, Resultat wird angezeigt	OK
Java Cryptography Extension Unlimited Strength Jurisdiction Policy Files sind installiert	Teilschlüssel können zusammengesetzt und Stimme entschlüsselt werden	Teilschlüssel können zusammengesetzt und Stimme entschlüsselt werden	OK
Eine getätigte Abstimmung wird ausgezählt. Es wurden keine Vote Records empfangen	Fehlermeldung im Log und Meldung, dass das Auszählen fehlgeschlagen ist	Fehlermeldung im Log und Meldung, dass das Auszählen fehlgeschlagen ist	OK

Tabelle 6.6: Auszählen der Stimmen

6.5.3 Zusammenfassung der Tests

Die Funktionstests in Bezug auf die Testbereiche Voter und Tallier haben gezeigt, dass die definierten Funktionen erfüllt werden. Während den Funktionstests mussten keine Fehler identifiziert werden. Nach dem erfolgreichen Überprüfen der definierten Testfälle ist jedoch noch ein Fall aufgetaucht, welcher nicht durch die Tests abgedeckt war. Bei der Eingabe eines Benutzernamens mit Umlauten auf einem anderen System, konnte sich der Benutzer nicht erfolgreich identifizieren. Dies ist wohl auf Probleme beim Berechnen der Prüfsumme mit einem Benutzernamen in welchem Umlaute vorkommen zurückzuführen. Aus Zeitgründen wurde dieses Problem dadurch gelöst, dass keine Benutzernamen mit Umlauten vergeben wurden. Gesamthaft betrachtet unterstreichen die Funktionstests jedoch, dass das System in einer guten Qualität implementiert werden konnte.

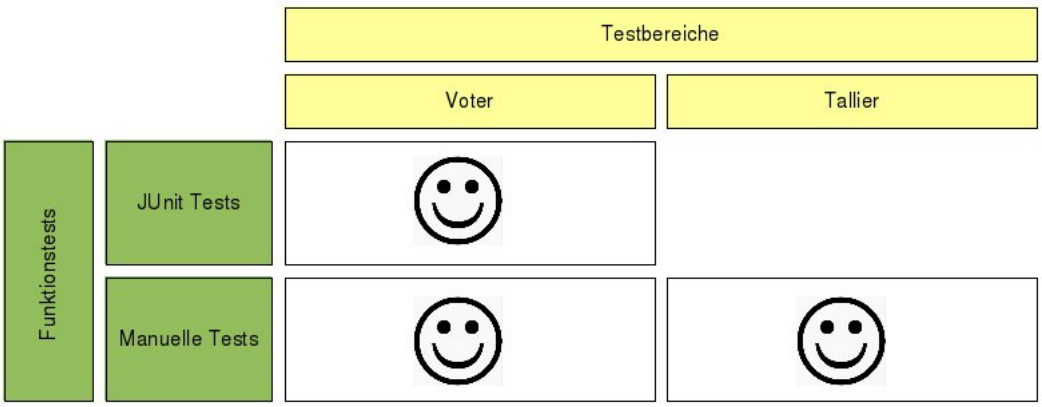

Abbildung 6.3: Zusammenfassung der Tests

Kapitel 7

Besprechung

7.1 Zielerreichung

Die Ziele der Arbeit wurden erreicht. Es konnten zwei für eine Demonstration am Swiss E-Voting Workshop 2009 taugliche Applikationen implementiert werden. Die Applikationen trugen dazu bei, den Teilnehmern des Workshops eine Vorstellung von einem transparenten E-Voting System zu vermitteln.

Das sekundäre Ziel der Mehrsprachigkeit, welches erst im Verlaufe der Realisierung definiert wurde, konnte ebenfalls umgesetzt werden. Aus zeitlichen Gründen, konnte die Kanalsicherung nicht wie gewünscht realisiert werden. In Absprache mit den Betreuern unserer Arbeit, wurde zugunsten der Stabilität auf einen zweiten Release mit Kanalsicherung verzichtet. Die dadurch gewonnene zeitliche Flexibilität wurde vor allem zugunsten von Detailarbeiten an der Benutzeroberfläche und Integrationstests eingesetzt. Es wurde jedoch eine Case Study zum Thema Kanalsicherung durchgeführt (siehe Anhang C), welche zum Schluss kommt, dass die Kanalsicherung relativ einfach nachträglich zu implementieren ist.

7.2 Lessons Learned

7.2.1 Vorgehensmodell

Als Vorgehensmodell wurde das Unified Process (UP) gewählt. Es hat sich gezeigt, dass sich dieses Vorgehensmodell für ein Projekt in diesem Rahmen bewährt. Für die Implementation der Appliaktionen hatten wir geplant, nach den Prinzipien von Test-driven development (TDD) vorzugehen. Dies stellte sich jedoch als sehr anspruchsvoll heraus und wir konnten dies leider nicht über die gesamte Phase der Realisierung wie gewollt umsetzen. Im Laufe der Implementierung sind wir, wohl auch aufgrund von zeitlichen Faktoren, von diesem Prinzip abgewichen. Wir haben vermehrt Klassen zuerst realisiert und danach mit JUnit Tests getestet. Grundsätzlich haben wir während dieser Arbeit jedoch sehr gute Erfahrungen mit TDD gemacht und werden in Zukunft sicherlich wieder versuchen mit TDD zu arbeiten.

7.2.2 Beurteilung der Phasen

Preparation Im Rahmen dieser Phase haben wir uns mit dem TrustVote-Protokoll auseinandergesetzt. Ausserdem haben wir erste Artefakte erstellt.

Ein fundiertes Wissen über das Protokoll, welches es galt umzusetzen war für diese Arbeit unumgänglich. Wir haben uns in dieser Phase vor allem viel Wissen angeeignet. Leider haben wir es ein wenig versäumt, während dieser Zeit auch klare Rahmenbedingungen für die Arbeit auszuarbeiten. Wir haben beispielsweise bereits mit dem Erstellen von Artefakten begonnen, ohne abzuklären ob diese auch wirklich nötig sind. Frühe und genaue Abklärungen betreffend den abzuliefernden Dokumenten sowie ein Zeitplan helfen sicherlich bei der Organisation des Projekts.

Design Im Rahmen dieser Phase erfolgte das Design der einzelnen Systemkomponenten. Weiter erstellten wir die Klassendiagramme und einen Prototyp der Benutzeroberfläche.

Die Desing Phase dauerte länger als geplant. Wir bekundeten Mühe, mit dem Erstellen von Sequenz- und Klassendiagrammen. Dadurch, dass wir seit längerem keine Programmierarbeit mehr geleistet hatten, ist es uns schwer gefallen, die benötigten Artefakte innerhalb der geplanten Zeit zu erstellen.

Positiv hervorzuheben ist, dass es uns während dieser Zeit gelungen ist, einen Prototyp der Benutzeroberfläche der Tallier Applikation zu erstellen. Im Allgemeinen jedoch, haben wir die Zeit, welche nötig ist um eine Benutzeroberfläche zu erstellen, unterschätzt. Die Zeit reichte nicht, um auch einen Prototyp der Voter Applikation zu erstellen. Ausserdem haben wir es versäumt, die Benutzeroberfläche in das Design mit einzubeziehen. Es hätte uns die Umsetzung erheblich erleichtert, wenn wir der Oberfläche bei der Erstellung unserer Artefakte mehr Beachtung geschenkt hätten. So mussten wir bei der Implementierung des GUI mehr Zeit aufwenden als geplant. Auch waren wir uns zu diesem Zeitpunkt noch nicht über die Grundsätze des GUI Designs im Klaren. In dieser Hinsicht wäre wohl eine genauere Recherche zum Thema in der Preparation Phase angebracht gewesen.

Trotz der Schwierigkeiten, welche beim Design auftraten, sind wir der Ansicht, dass sich der Mehraufwand ausgezahlt hat. Durch das genaue Design konnte in der Phase der Realisierung vermehrt Zeit eingespart werden. Dies führte wiederum dazu, dass wir uns vermehrt auch um Detailarbeiten an der Benutzeroberfläche kümmern konnten.

Realisierung Im Rahmen der Realisierung wurde die Software entwickelt und getestet.

Bei der Realisierung zeigte sich nun, dass sich der Mehraufwand, welcher in der Phase des Designs betrieben wurde, auszahlte. Trotzdem tauchten einige Schwierigkeiten auf. Es kristallisierten sich einige Härtefälle in Bezug auf die Implementierung der kryptographischen Anforderungen heraus. Zudem zeigte sich bei der Umsetzung der Anforderungen des Protokolls, dass sich Teile davon, wie beispielsweise das

Threshold Blind Signature Verfahren, nicht wie geplant implementieren liessen. Dies brachte partielle Verzögerungen mit sich. Auch im Rahmen der Implementierung der Benutzeroberfläche tauchten Probleme auf. Einige hätten wohl mit einer besseren Planung vermieden werden können, andere waren jedoch auch auf unseren bis anhin geringen Erfahrungsschatz im Bereich der GUI Programmierung zurückzuführen. Hinzu kamen einige Missverständnisse im Bereich der Gestaltung der Benutzeroberfläche. Deren Beseitigung nahm zusätzlich Zeit in Anspruch. Unklarheiten in bei der Gestaltung der Benutzeroberfläche hätten wohl mit einem Prototypen verhindert oder wenigsten auf ein Minimum reduziert werden können.

Gegen Ende der Phase ging es zusätzlich darum, die Zusammenarbeit unserer Software mit den restlichen, im Rahmen des TrustVote Projekts zu realisierenden Komponenten des Protokolls sicherzustellen. Es sollte sich zeigen, dass dies wohl die grösste Herausforderung in der Realisierung werden sollte. Gerade die Implementierung des Threshold Blind Signature Verfahrens nahm relativ viel Zeit in Anspruch. Auch gab es organisatorische Schwierigkeiten zu lösen. Teilweise verzögerte sich die Implementierung von Komponenten der Applikationen aufgrund dessen, dass benötigte Webservices noch nicht oder nur partiell zur Verfügung standen.

7.2.3 Projektmanagement und Organisation

Das Arbeiten im Team stellt sicherlich eine Herausforderung an alle Beteiligten dar. Durch die Tatsache, dass die Arbeit während dem Semester durchgeführt wurde, erforderte aufgrund von unterschiedlichen Modulbelegungen teilweise einigen Organisationsaufwand. Wir haben versucht, uns so oft als möglich auszutauschen, damit beide Teammitglieder, auch im Fall eines krankheitsbedingten Ausfalls, immer auf dem aktuellsten Informationsstand waren. Aufgrund dessen, dass die restlichen Komponenten des TrustVote-Protokolls von einem Master-Studierenden realisiert wurden, welcher teilweise Vorlesungen in Zürich hatte, kam es vor, dass Probleme, welche die Infrastruktur der Webservices betrafen, nicht von uns selbst gelöst werden konnten. Dies hat partiell zu einigen, jedoch nicht gravierenden Verzögerungen geführt.

Die Zusammenarbeit im Team wurde als sehr positiv empfunden. Es ermöglicht einen hohen Lerneffekt und wirkt motivierend. Ausserdem konnten die Arbeitsergebnisse einfach überprüft und gegebenenfalls verbessert werden.

7.3 Mögliche Erweiterungen

Durch die Verwendung von Bouncy Castle Lightweight API anstelle des Bouncy Castle Java Cryptography Extension (JCE) Providers für die Umsetzung von kryptographischen Funktionen, könnten die Applikationen den Benutzern einfacher zur Verfügung gestellt werden, da die Java Cryptography Extension Unlimited Strength Jurisdiction Policy Files nicht installiert werden müssten.

Eine weitere Erweiterung könnte ein detaillierte Hilfe oder Wegleitung für den Benutzer sein. Zudem wäre es wünschenswert, dass die Benutzer die von ihnen bevor-

zugte Sprache selber auswählten können. Dies würde das Verteilen der Applikationen vereinfachen, da jeweils nur eine, mehrsprachige Version, erstellt werden müsste.

Momentan werden von den Voter und Tallier Applikationen AES, RSA und SHA-1 unterstützt. In Zukunft könnte Unterstützung für weitere Verschlüsselungsverfahren, wie beispielsweise 3DES, implementiert werden.

Ausserdem könnte durch Verwendung des ConfigurationContexts bei allen Web Service aufrufen die Flexibilität gesteigert werden. Bei einer Änderung der URL des Web Services müsste somit die Web Service Referenz in der betroffenen Applikation nicht verändert werden.

Die Applikationen könnten zudem um eine Log-Datei erweitert werden, in welche Fehler- und Statusmeldungen geschrieben werden. Dies würde die Fehlersuche und den Support bei einer allfällig fehlerhaften Funktionsweise erleichtern.

7.4 Ausblick

Die Voter und Tallier Applikationen wurden klar für einen Showcase des von der Berner Fachhochschule entwickelten TrustVote-Protokolls entwickelt. E-Voting wird in Zukunft immer mehr an Bedeutung und Attraktivität gewinnen. Mit Hilfe der von uns entwickelten Applikationen lässt sich der mögliche Ablauf einer elektronischen Abstimmung auch für Laien verständlich darstellen. Die Pilotprojekte zur elektronischen Stimmabgabe in Genf, Neuenburg und Zürich zeigen, dass E-Voting durchaus eine Ergänzung zur Stimmabgabe an der Urne und zur brieflichen Stimmabgabe sein kann. Wie das Beispiel der Wahl zur Hochschülerschaftsvertretung (ÖH) in Östereich diesen Mai gezeigt hat, ist es jedoch noch ein weiter Weg bis zur Einführung eines E-Voting Systems auf Bundesebene. Allem voran wird die Forderung nach Transparenz und Offenheit beim E-Voting ein sehr wichtiger Faktor für eine Einführung sein. Zudem ist stellt sich die Frage, ob es gelingt, sicherzustellen, dass die Wähler tatsächlich Vertrauen in ein E-Voting System haben. Angesichts der technischen Komplexität kann dies eine enorm schwierige Aufgabe sein, zu deren Lösung, so hoffen wir, wir mit der Realisierung der Voter und Tallier Applikationen einen Beitrag leisten konnten.

Anhang A

Funktionale Anforderungen

A.1 Benutzerkreise (Actors)

Für die verschiedenen Subsysteme existieren aus Benutzersicht verschiedene Benutzerrollen. In den nachfolgend aufgeführten Use-Cases wird auf diese Rollen Bezug genommen.

A.1.1 Voter

Der Actor Voter wird in Tabelle A.1 beschrieben.

Actor	Voter
Typ	Primary Actor
Beschreibung	Der Voter kann einen Stimmzettel ausfüllen und absenden. Nach dem Senden der Stimme kann der Voter diese auf dem Public Board kontrollieren und wenn nötig mit einer zweiten Stimme, welche er auf dem Postweg abgibt, revozieren. Sobald der Voter die elektronische Stimme abgegeben hat, kann der Prozess nicht mehr abgebrochen oder rückgängig gemacht werden.
Zweck	-

Tabelle A.1: Actor: Voter

A.1.2 Tallier

Der Actor Tallier wird in Tabelle A.2 beschrieben.

Actor	Tallier
Typ	Primary Actor
Beschreibung	Wenn alle Stimmen abgegeben und vollständig auf dem Public Board publiziert sind, zählt der Tallier die Stimmen und stellt das erhaltene Resultat in Diagrammen dar.
Zweck	Auszählen einer Wahl oder Abstimmung.

Tabelle A.2: Actor: Tallier

A.1.3 Authority

Der Actor Authority wird in Tabelle A.3 beschrieben.

Actor	Authority
Typ	Supporting Actor
Beschreibung	Die Authority stellt sicher, dass ein Voter zur Stimme berechtigt ist. Wenn dies der Fall ist, wir seine Stimme von der Authority signiert.
Zweck	Stellt sicher, dass Voter nicht wählen oder abstimmen können, wenn sie nicht dazu berechtigt sind.

Tabelle A.3: Actor: Authority

A.1.4 Public Board

Der Actor Public Board wird in Tabelle A.4 beschrieben.

Actor	Public Board
Typ	Supporting Actor
Beschreibung	Die ausgefüllten Stimmzettel werden von den Votern an das Public Board gesendet. Nach Abschluss der Wahl oder Abstimmung werden zudem die Teilschlüssel zu jeder Stimme auf dem Public Board veröffentlicht.
Zweck	Voter können sicherstellen, dass Ihre Stimme korrekt abgegeben wurde.

Tabelle A.4: Actor: Public Board

A.1.5 Registration Board

Der Actor Authority wird in Tabelle A.5 beschrieben.

Actor	Registration Board
Typ	Supporting Actor
Beschreibung	Jeder Voter sendet seine Voter ID und einen Hashwert seiner Credentials an das Registration Board. Die Authorities greifen auf diese Einträge zu und stellen sicher, dass nur authorisierte Voter auch wirklich an der Wahl oder Abstimmung teilnehmen dürfen.
Zweck	Stellt sicher, dass kein Stimmender zweimal einen Stimmzettel abgeben kann.

Tabelle A.5: Actor: Registration Board

A.1.6 Key Collector

Der Actor Key Collector wird in Tabelle A.6 beschrieben.

Actor	Key Collector
Typ	Supporting Actor
Beschreibung	Jeder Voter sendet einen Teil des Schlüssels, mit welchem er den Stimmzettel verschlüsselt hat, an einen Key Collector. Die Teilschlüssel werden am Ende einer Wahl oder Abstimmung veröffentlicht und dem Public Board zugänglich gemacht.
Zweck	Sammelt die Teilschlüssel und stellt sicher, dass diese nicht vor Ende der Abstimmung bekannt werden.

Tabelle A.6: Actor: Key Colletors

A.1.7 Administration

Der Actor Administration wird in Tabelle A.7 beschrieben.

Actor	Administration
Typ	Supporting Actor
Beschreibung	Administriert einen Voting Event
Zweck	Veröffentlicht den leeren Stimmzettel und definiert die zur Wahl oder Abstimmung berechtigten Voter.

Tabelle A.7: Actor: Administration

A.2 Voter

A.2.1 UC1: Stimme abgeben

Bereich	Voter Anwendung
Level	User goal
Primary Actor	Voter
Beteiligte Actors	Am Use-Case sind mehrere Actors beteiligt: • Administration • Registration Board • Authorities • Public Board • Key Collectors
Vorbedingungen	Der Voter hat die Voter Anwendung installiert und ausserdem sind folgende Bedingungen erfüllt: • Die Voter Anwendung kann auf die Public Keys der Authorities zugreifen • Der Voter Anwendung sind die URLs zu den Web Services aller beteiligten Actors bekannt • Der Voter hat einen Benutzernamen, ein Passwort und die zugehörige Prüfsumme auf dem Postweg erhalten • Es existiert ein leerer Stimmzettel, welcher bei der Administration abgerufen werden kann
Ergebnis	Der ausgefüllte Stimmzettel wird korrekt versendet und auf dem Public Board veröffentlicht

Ablauf

Ausfüllen und versenden eines Stimmzettles:

1. Der Voter startet die Voter Applikation
2. Der Voter indentifiziert sich mit den auf dem Postweg erhaltenen Daten und gelangt mit "Weiter" zum leeren Stimmzettel
3. Der Stimmzettel wird ausgefüllt
4. Eine Zusammenfassung zeigt eine Übersicht über die gewählten Kandidierenden und die Anworten zu den Abstimmungen
5. Mit "Senden" wird der Stimmzettel versendet
6. Der Stimm-Kennwert und das Stimm-Geheimnis werden angezeit und der Voter informiert, dass die Stimme erfolgreich versendet wurde

Varianten

Mögliche Abweichungen vom Standard-Ablauf:

- Die Anwendung oder das zugrundeliegende Betriebssystem stürzt ab
  - Die Daten sind verloren. Der Stimmzettel muss erneut ausgefüllt werden. Wenn die Stimme noch nicht versendet wurde, kann diese erneut versendet werden
- Der Voter beendet die Applikation während des Ausfüllens des Stimmzettels
  - Dem Voter wird ein Hinweis angezeigt, dass beim Beenden der Anwendung alle bisher vorgenommenen Änderungen verloren gehen

- Der Voter beendet die Applikation während des Verarbeitens der Stimme
  - Dem Voter wird ein Hinweis angezeigt, dass die Anwendung während der Verarbeitung der Stimme nicht beendet werden kann
- Der Anwendung ist es nicht möglich den leeren Stimmzettel bei der Administration abzurufen
  - Eine Fehlermeldung wird an den Benutzer ausgegeben und die Anwendung muss beendet werden
- Der von der Administration erhaltene Stimmzettel enthält keine Abstimmungen oder Wahlen
  - Eine Fehlermeldung wird an den Benutzer ausgegeben und die Anwendung muss beendet werden
- Der vom Voter eingegebene Benutzername und Passwort stimmen nicht mit der Prüfsumme überein
  - Eine Hinweis wird an den Benutzer ausgegeben. Solange die Angaben nicht korrekt eingegeben wurde, kann der Stimmzettel nicht ausgefüllt werden
- Der Stimmzettel kann nicht versendet werden, weil während dem Verarbeiten der Stimme ein Fehler auftritt
  - Eine entsprechende Fehlermeldung wird im Statuslog ausgegeben und der Voter informiert, dass das Verarbeiten der Stimme fehlgeschlagen ist.
  - Der Voter muss seinen Stimmzettel mit dem erhaltenen Stimm-Geheimnis über den Postweg versenden

Tabelle A.8: UC1: Voter Anwendung

A.3 Tallier

A.3.1 UC2: Stimmen auszählen

Bereich	Tallier Anwendung
Level	User goal
Primary Actor	Tallier
Beteiligte Actors	Public Board
Vorbedingungen	Die Tallier Anwendung ist installiert und ausserdem sind folgende Bedingungen erfüllt: • Die Tallier Anwendung kann auch die Public Keys der Authorities zugreifen • Der Tallier Anwendung sind die URLs zu den Web Services aller beteiligten Actors bekannt • Es existiert ein leerer Stimmzettel, welcher bei der Administration abgerufen werden kann • Die Teilschlüssel wurden von den Key Collectors veröffentlich und vom Public Board verarbeitet
Ergebnis	Die ausgefüllten Stimmzettel werden ausgezählt und die Ergebnisse publiziert
Ablauf	Auszählen einer Wahl oder Abstimmung 1. Die Tallier Applikation wird gestartet 2. Mit "Auszählen" wir die Auszählung gestartet 3. Ist die Auszählung beendet, können mit "Anzeigen" die Resultate angezeigt werden

Varianten	Mögliche Abweichungen vom Standard-Ablauf: • Die Anwendung oder das zugrundeliegende Betriebssystem stürzt ab – Die Daten sind verloren. Die Wahl oder Abstimmung muss erneut ausgezählt werden • Die Applikation wird beendet – Es wird ein Hinweis angezeigt, dass beim Beenden der Anwendung alle bisher vorgenommenen Änderungen verloren gehen • Der Anwendung ist es nicht möglich den leeren Stimmzettel von der Administration zu erhalten – Eine Fehlermeldung wird an den Benutzer ausgegeben und die Anwendung muss beendet werden • Der von der Administration erhaltene Stimmzettel enthält keine Abstimmungen oder Wahlen – Eine Fehlermeldung wird an den Benutzer ausgegeben und die Anwendung muss beendet werden • Die Wahl oder Abstimmung kann nicht ausgezählt werden, weil während dem Verarbeiten der Stimmen ein Fehler auftritt – Eine entsprechende Fehlermeldung wird im Statuslog ausgegeben und der Benutzer informiert, dass das Auszählen der Stimmen fehlgeschlagen ist. – Der Benutzer muss sich bei seiner Administration melden

Tabelle A.9: UC2: Tallier Anwendung

Anhang B

Klassendiagramme

B.1 Einleitung

Im Folgenden werden die Klassendiagramme der Voter und Tallier Applikationen detailliert beschrieben und dargestellt. Zu beachten ist, dass hellblaue Pakete mit violetten Klassen im Rahmen dieses Projekts realisiert wurden. Alle anderen Elemente, welche in einer anderen Farbe dargestellt werden, standen zur Verfügung und mussten nicht selbst implementiert werden. Abschnitt B.2 beschreibt die Klassen der Voter, Abschnitt B.3 diejenigen der Tallier Applikation.

B.2 Klassendiagramme Voter

Paket ch.bfh.trustvote.showcase.voter.gui

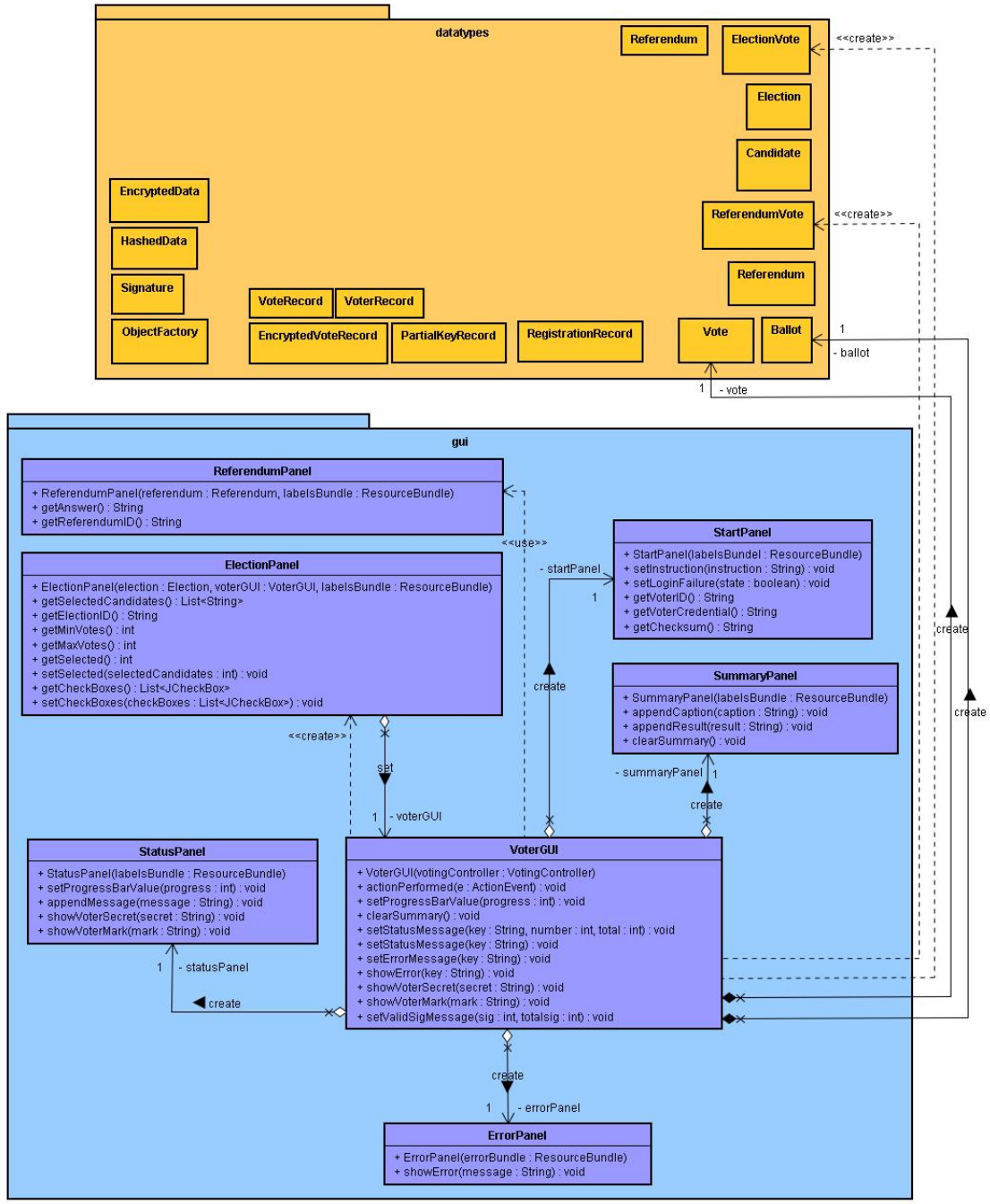

Abbildung B.1: Klassendiagram des Pakets `voter.gui`

B Klassendiagramme

Paket	Klasse	Beschreibung
gui	VoterGUI	Das Voter GUI repräsentiert den Dialog, durch welchen ein Stimmender für das Ausfüllen des Stimmzettels geführt wird.
	StatusPanel	Das Status Panel wird beim Verarbeiten der Stimme angezeigt. In diesem Panel werden auch Statusmeldungen für den Benutzer ausgegeben.
	ErrorPanel	Wird angezeigt, wenn ein schwerwiegender Fehler beim Start der Applikation auftritt.
	SummaryPanel	Das Summary Panel präsentiert dem Stimmenden die von ihm getroffenen Entscheidungen.
	StartPanel	Das Start Panel wird beim Start der Applikation angezeigt. Es verlangt vom Benutzer sich zu authentifizieren bevor er mit dem Ausfüllen des Stimmzettels fortfahren kann.
	ReferendumPanel	Für jede Abstimmung existiert ein Referendum Panel. Es erlaubt dem Benutzer seine Antwort auf eine Abstimmungsfrage abzugeben.
	ElectionPanel	Für jede Wahl existiert ein Election Panel. Es erlaubt dem Benutzer die gewünschten Kandidierenden einer Wahl auszuwählen.
datatypes		Siehe Tabelle 4.1.

Tabelle B.1: Klassenbeschreibung des Pakets `voter.gui`

Paket ch.bfh.trustvote.showcase.voter.controller

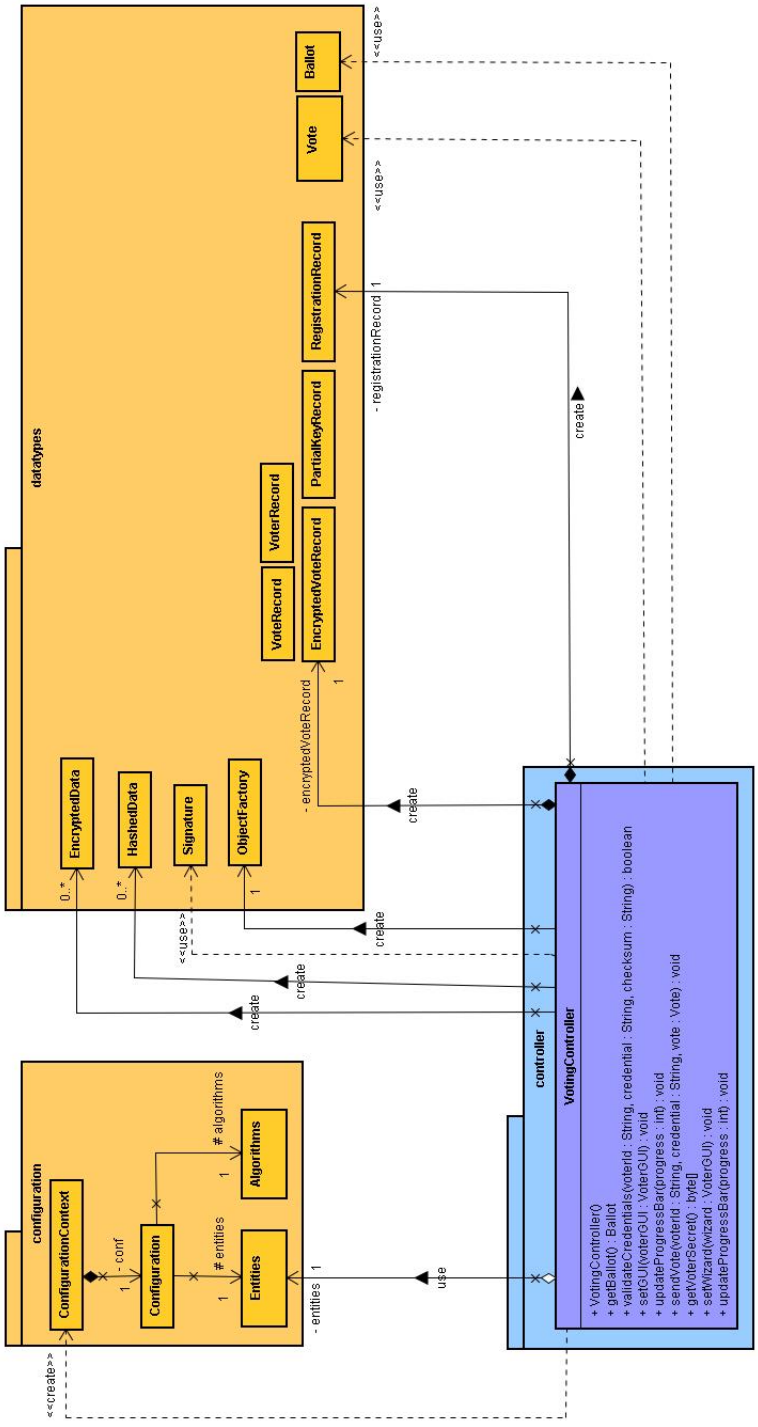

Abbildung B.2: Klassendiagram des Pakets `voter.controller`

Paket	Klasse	Beschreibung
controller	VotingController	Die Klasse VotingController, aus hierarchischer Sicht die erste Instanz nach dem GUI, ist die Schaltzentrale zwischen UI-Layer und den restlichen Layer. Sie erledigt selbst keine Arbeit sondern delegiert Befehle vom GUI weiter zum Domain-Layer und schickt Informationen, die am GUI ausgegeben werden müssen zurück an den UI-Layer.
configuration		Siehe Tabelle 4.4.
datatypes		Siehe Tabelle 4.1.

Tabelle B.2: Klassenbeschreibung des Pakets `voter.controller`

Paket ch.bfh.trustvote.showcase.voter.connectionmanager

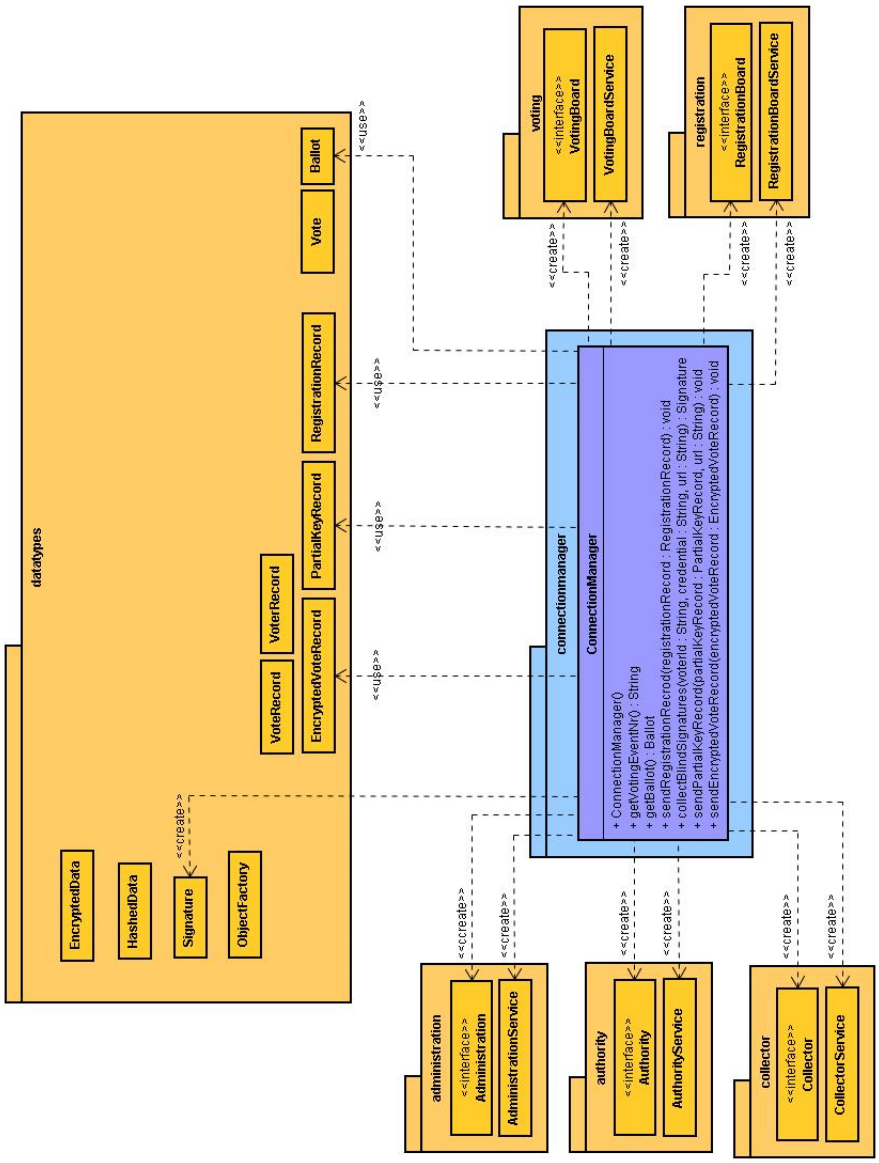

Abbildung B.3: Klassendiagram des Pakets `voter.connectionmanager`

Paket	Klasse	Beschreibung
connection-manager	ConnectionManager	Die Klasse ConnectionManager implementiert die Verbindungen zu den Webservices. Sämtliche Daten für und von den Webservices werden über die Methoden seiner Instanz abgearbeitet.
datatypes		Siehe Tabelle 4.1.

Tabelle B.3: Klassenbeschreibung des Pakets `voter.connectionmanager`

Paket ch.bfh.trustvote.showcase.voter.cryptomanager

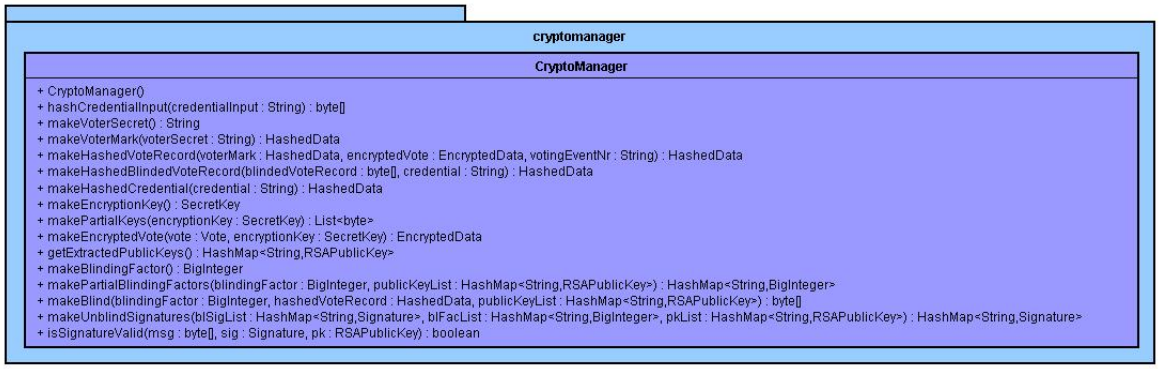

Abbildung B.4: Klassendiagram des Pakets `voter.cryptomanager`

Paket	Klasse	Beschreibung
crypto-manager	CryptoManager	Die Klasse CryptoManager ist verantwortlich für sämtliche kryptographischen Operationen, die im Application-Layer aufgerufen werden. Sie leitet die Befehle weiter an die betreffenden Klassen im Service-Layer.

Tabelle B.4: Klassenbeschreibung des Pakets `voter.cryptomanager`

Paket ch.bfh.trustvote.showcase.voter.crypto

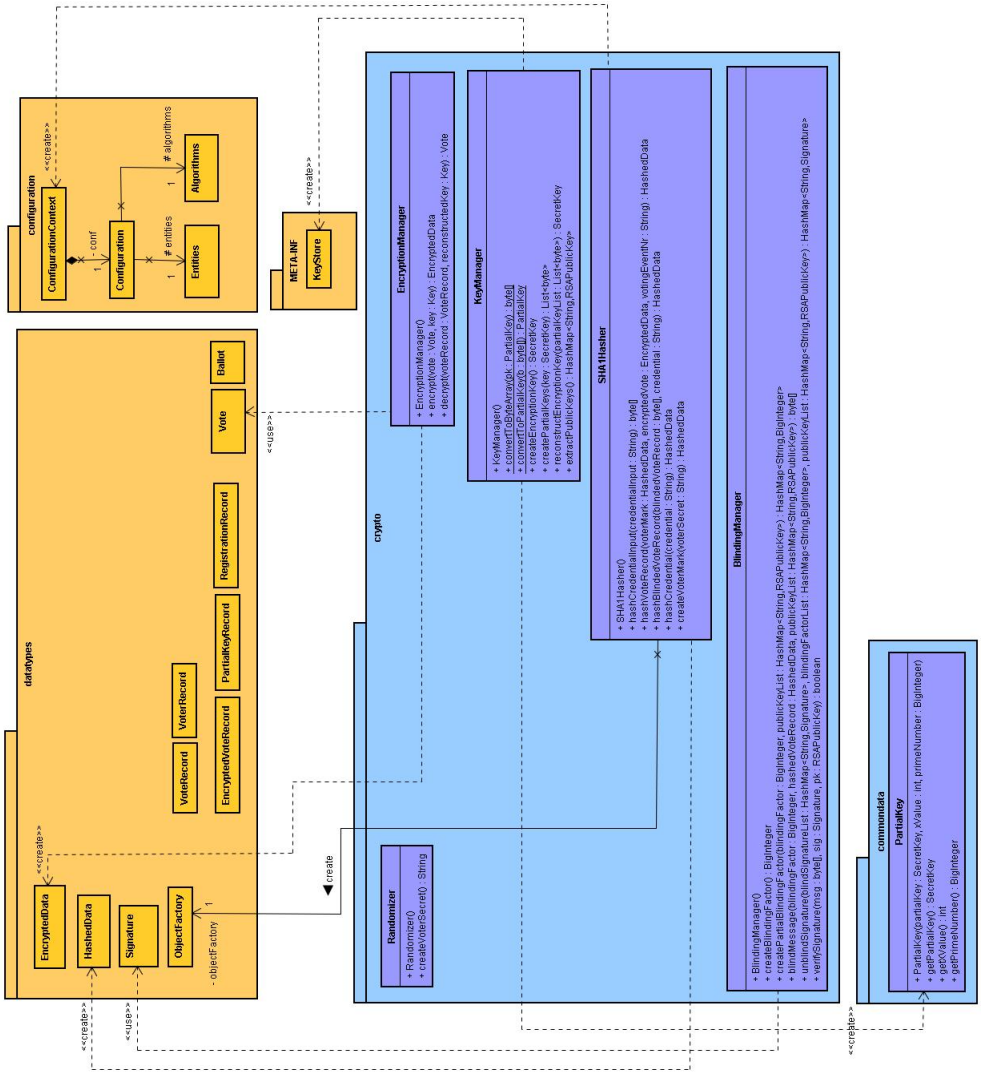

Abbildung B.5: Klassendiagramm des Pakets `voter.crypto`

Paket	Klasse	Beschreibung
crypto	Randomizer	Erstellt für die abstimmende Person ein zufälliges VoterSecret aus sechs Base64 codierten Zeichen.
	EncryptionManager	Verschlüsselt und entschlüsselt Vote-Objekte mit dem Block-Cipher AES im CTR-Mode mit Schlüssellänge 256 Bit.
	KeyManager	Ist für sämtliche Schlüssel-Objekte verantwortlich. Sie generiert den geheimen Schlüssel, um das Vote-Objekt zu verschlüsseln, unterteilt den geheimen Schlüssel in Teilschlüssel für die Collector-Webservices, rekonstruiert aus den Teilschlüsseln den geheimen Schlüssel und entpackt die Public Keys aus dem Java KeyStore.
	SHA1Hasher	Bildet den SHA1-Hashwert von Daten-Objekten.
	BlindingManager	Implementiert in seinen Methoden die Algorithmen des Threshold Blind Signature Schemas. Konkret heisst das, mit Hilfe dieser Klasse wird der Blindierungsfaktor erstellt sowie die Blindierungsfaktoren für jede Authoroty, werden Daten blind gemacht, Signaturen entblindet und verifiziert.
commondata	PartialKey	Enthält die notwendigen Daten, um aus einer gewissen Anzahl Teilschlüsseln wieder den geheimen Schlüssel konstruieren zu können.
configuration		Siehe Tabelle 4.4.
datatypes		Siehe Tabelle 4.1.
META-INF	KeyStore	Ein KeyStore-Objekt wird verwendet, um aus dem Java KeyStore `keystore.jks` die Public Keys der Authoroties zu entpacken.

Tabelle B.5: Klassenbeschreibung des Pakets `voter.crypto`

B.3 Klassendiagramme Tallier

Paket ch.bfh.trustvote.showcase.tallier.gui

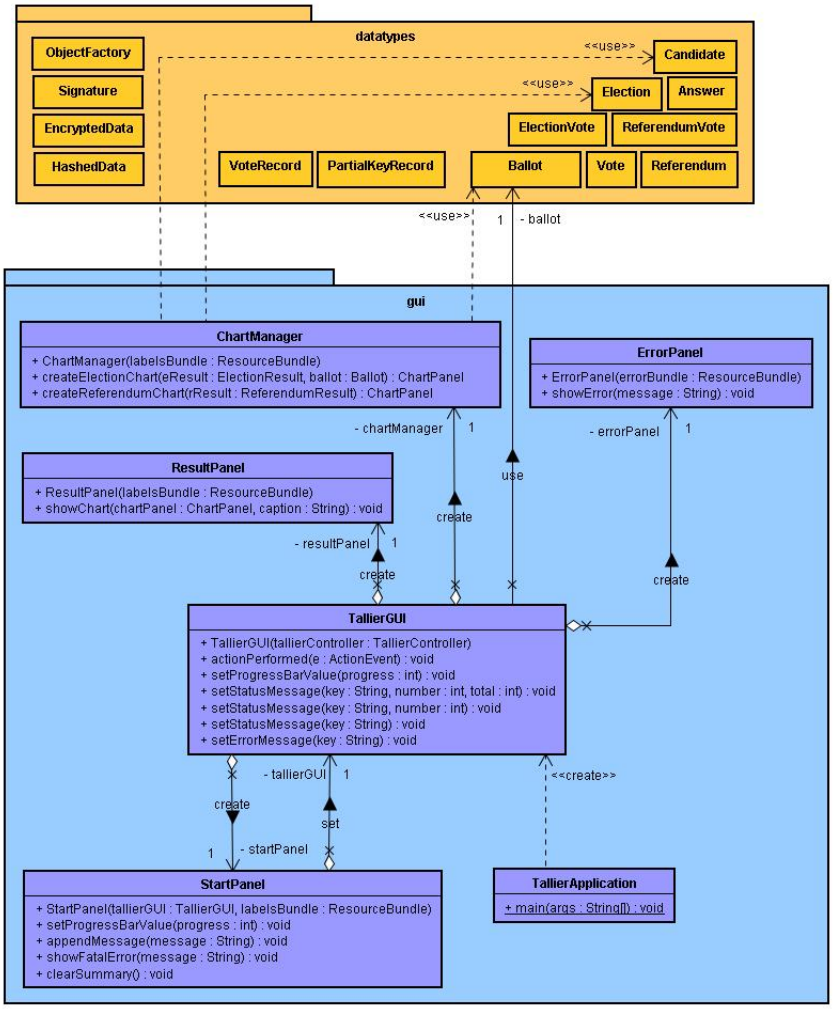

Abbildung B.6: Klassendiagram des Pakets `tallier.gui`

Paket	Klasse	Beschreibung
gui	VoterGUI	Das Tallier GUI repräsentiert den Dialog, durch welchen ein Benutzer für das Auszählen einer Wahl oder Abstimmung geführt wird.
	ChartManager	Der ChartManager ist für das Erstellen der Diagramme aus den Resultaten der Wahlen oder Abstimmungen zuständig.
	ErrorPanel	Wird angezeigt, wenn ein schwerwiegender Fehler beim Start der Applikation auftritt.
	ResultPanel	Im Result Panel werden die Resultate der Abstimmungen oder Wahlen angezeigt.
	StartPanel	Das Start Panel wird beim Start der Applikation angezeigt. In diesem Panel werden auch Statusmeldungen für den Benutzer ausgegeben.
datatypes		Siehe Tabelle 4.1.

Tabelle B.6: Klassenbeschreibung des Pakets `tallier.gui`

Paket ch.bfh.trustvote.showcase.tallier.controller

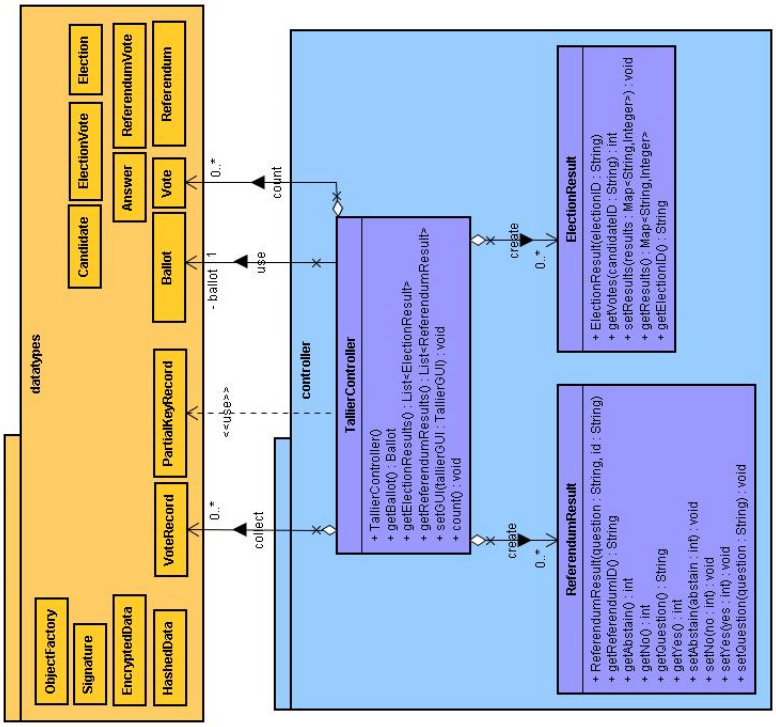

Abbildung B.7: Klassendiagram des Pakets `tallier.controller`

Paket	Klasse	Beschreibung
controller	TallierController	Die Klasse TallierController, aus hierarchischer Sicht die erste Instanz nach dem GUI, ist die Schaltzentrale zwischen UI-Layer und den restlichen Layer. Sie erledigt selbst keine Arbeit sondern delegiert Befehle vom GUI weiter zum Domain-Layer und schickt Informationen, die am GUI ausgegeben werden müssen zurück an den UI-Layer.
configuration		Siehe Tabelle 4.4.
datatypes		Siehe Tabelle 4.1.

Tabelle B.7: Klassenbeschreibung des Pakets `tallier.controller`

Paket ch.bfh.trustvote.showcase.tallier.connectionmanager

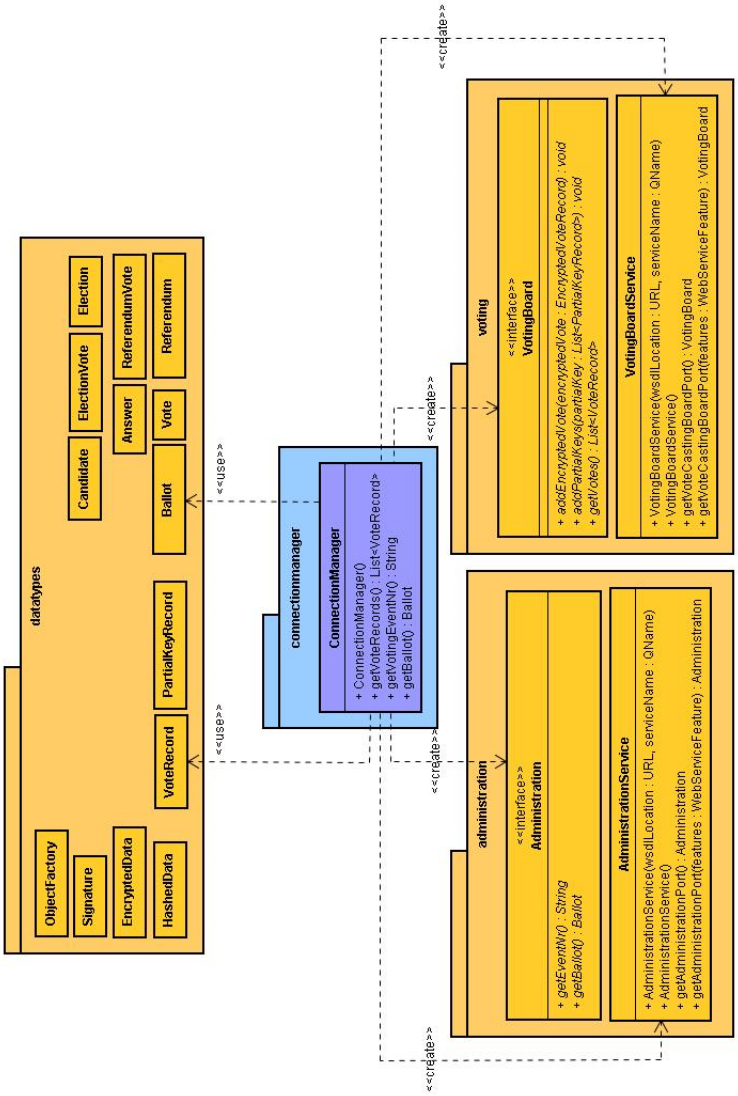

Abbildung B.8: Klassendiagram des Pakets `tallier.connectionmanager`

Paket	Klasse	Beschreibung
connection-manager	ConnectionManager	Die Klasse ConnectionManager implementiert die Verbindungen zu den Webservices. Sämtliche Daten für und von den Webservices werden über die Methoden seiner Instanz abgearbeitet.
datatypes		Siehe Tabelle 4.1.

Tabelle B.8: Klassenbeschreibung des Pakets `tallier.connectionmanager`

Paket ch.bfh.trustvote.showcase.tallier.cryptomanager

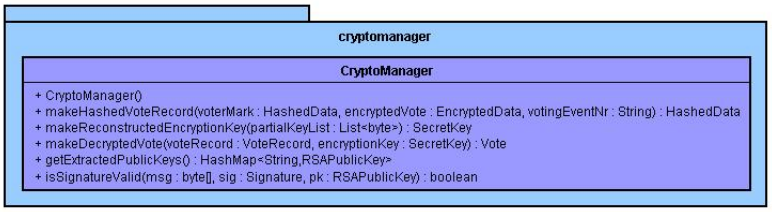

Abbildung B.9: Klassendiagram des Pakets `tallier.cryptomanager`

Paket	Klasse	Beschreibung
crypto-manager	CryptoManager	Die Klasse CryptoManager ist verantwortlich für sämtliche kryptographischen Operationen, die im Application-Layer aufgerufen werden. Sie leitet die Befehle weiter an die betreffenden Klassen im Service-Layer.

Tabelle B.9: Klassenbeschreibung des Pakets `tallier.cryptomanager`

Paket ch.bfh.trustvote.showcase.voter.crypto

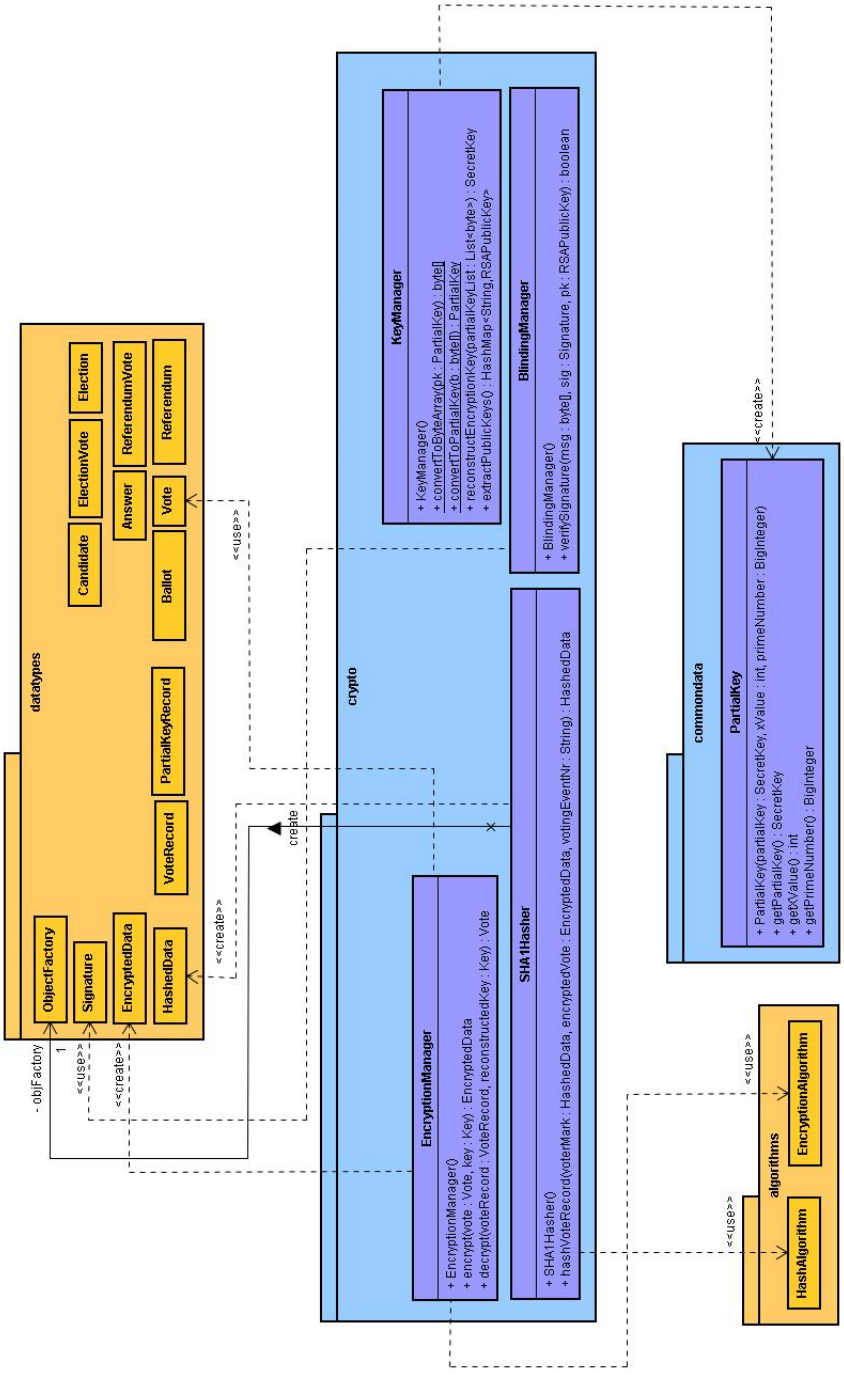

Abbildung B.10: Klassendiagram des Pakets `tallier.crypto`

Paket	Klasse	Beschreibung
	EncryptionManager	Verschlüsselt und entschlüsselt Vote-Objekte mit dem Block-Cipher AES im CTR-Mode mit Schlüssellänge 256 Bit.
	KeyManager	Ist für sämtliche Schlüssel-Objekte verantwortlich. Die Klasse rekonstruiert aus den Teilschlüsseln den geheimen Schlüssel und entpackt die Public Keys aus dem Java KeyStore.
	SHA1Hasher	Bildet den SHA1-Hashwert von VoteRecord-Objekten zum Überprüfen der Signaturen.
	BlindingManager	Verifiziert die mit den Stimmen erhaltenen Signaturen.
common-data	PartialKey	Enthält die notwendigen Daten, um aus einer gewissen Anzahl Teilschlüsseln wieder den geheimen Schlüssel konstruieren zu können.
configuration		Siehe Tabelle 4.4.
datatypes		Siehe Tabelle 4.1.
META-INF	KeyStore	Ein KeyStore-Objekt wird verwendet, um aus dem Java KeyStore `keystore.jks` die Public Keys der Authoroties zu entpacken.

Tabelle B.10: Klassenbeschreibung des Pakets `tallier.crypto`

Anhang C

Kanalsicherung

C.1 Hintergrund

Die Sicherung der Kanäle zwischen den Voter und Tallier Applikationen sowie den Web Services, auf welche diese zugreifen, wurde als sekundäres Ziel dieser Arbeit definiert. Aus zeitlichen Gründen, konnte diese Kanalsicherheit jedoch nicht implementiert werden. Die Sicherung eines Kanals zwischen einem Web Service und einer Java Applikation, welche auf diesen Service zugreift wurde daher nur als Case Study umgesetzt. Abschnitt C.3 verdeutlicht die nötige Vorgehensweise zum Sichern eines Web Services mit Transport Security (SSL). Es wurde versucht eine Art Anleitung zu erstellen, welches die nötigen Schritte zum Sichern des Kanals möglichst genau und vor allem nachvollziehbar beschreibt.

C.2 Aufbau

Für die Case Study wurden zwei neue NetBeans Projekte erstellt. Zum einen ist das der benötigte Web Service, welchen es zu Sichern gilt, zum anderen ein Java Client, welcher auf diesen Service zugreift. Abblidung C.1 gibt einen Überblick über die zwei Projekte.

Die Case Study wurde auf einem Notebook mit GNU Linux durchgeführt. Pfadangaben können auf einem anderen System gegebenenfalls abweichen. Listing C.1 zeigt den Quellcode der Klasse `SecuredStockQuoter` des Webservers und Listing C.2 denjenigen der Klasse `Main` des Web Service Clients.

```
1  @WebService()
2  public class SecuredStockQuoter {
3
4      HashMap<String,Double> hashMap = null;
5
6      public SecuredStockQuoter() {
7          this.hashMap = new HashMap();
8          this.hashMap.put("IBM", 55.67);
9          this.hashMap.put("Microsoft", 129.45);
10     }
11     /**
12      * Web service operation
13      */
```

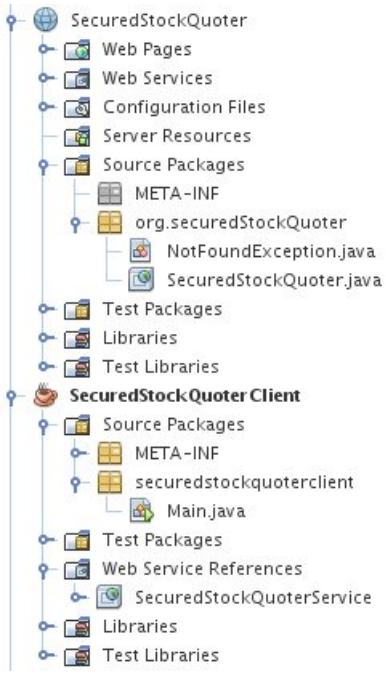

Abbildung C.1: Übersicht Projekte Case Study

```
14      @WebMethod(operationName = "getQuote")
15      public double getQuote(@WebParam(name = "symbol") String symbol) throws
            NotFoundException {
16          return this.hashMap.get(symbol);
17      }
18  }
```

Listing C.1: SecuredStockQuoter Web Service

```
1   public class Main {
2
3       /**
4        * @param args the command line arguments
5        */
6       public static void main(String[] args) {
7
8           try { // Call Web Service Operation
9               org.securedstockquoter.SecuredStockQuoterService service = new org
                    .securedstockquoter.SecuredStockQuoterService();
10              org.securedstockquoter.SecuredStockQuoter port = service.
                    getSecuredStockQuoterPort();
11              // TODO initialize WS operation arguments here
12              java.lang.String symbol = "IBM";
13              // TODO process result here
14              double result = port.getQuote(symbol);
15              System.out.println("Result = "+result);
16          } catch (Exception ex) {
17              System.out.println(ex.getMessage());
18          }
19      }
20  }
```

Listing C.2: SecuredStockQuoter Web Service Client

C.3 Vorgehen

Zu Testzwecken wurde der Web Service zuerst mit dem Sicherheitsmechanismus *Username Authentication with Symmetric Key* gesichert. Damit der Web Service Client mit Benutzernamen und Passwort auf den Web Service zugreifen konnte musste ein Benutzer zum File Realm von Glassfish hinzugefügt werden. Ausserdem musste dem Projekt die Glassfish Library webservices-rt.jar, welche unter *<Glassfish HOME>/lib* zu finden ist, hinzugefügt werden. Ohne diese Library tritt beim Verbinden zum gesichtern Web Service eine *javax.xml.ws.soap.SOAPFaultException* auf. In den Sicherheitseinstellungen des Web Service Clients muss ausserdem der Pfad zum Truststore, welcher das benötigte Server-Zertifikat enthält angegeben werden (Abbildung C.2). In dieser Case Study werden die Standard Keystores von Netbeans verwendet, in welchen selbstsignierte Zertifikate des Servers abgelegt sind. Die Keystores sind unter dem Pfad */home/<USERNAME>/.personalDomain/personalDomain/config/* zu finden.

Abbildung C.2: Konfiguration TrustStore des SecuredStockQuoter Service

Der SecuredStockQuoter Service wurde in einem nächsten Schritt für die Verwendung von SSL zur Sicherung des Web Services konfiguriert. Folgende Schritte sind dazu notwendig:

- In den Web Service Attributs muss der Sicherheitsmechanismus *Transport Security (SSL)* ausgewählt werden.
- Damit der Service das HTTPS Protokol benutzt, muss die Datei `web.xml` angepasst werden (Abbildungen C.3 und C.4):
  - Durch Doppelklicken von web.xml wird die Datei im Netbeans Editor geöffnet.
  - Das Security Tab muss ausgewählt werden.
  - Unter Security Constraints wird nun eine neue Security Constraint hinzugefügt.
  - In der neuen Security Constraint muss eine neue Web Resource Collection hinzugefügt werden.

  - Es muss sichergestellt werden, dass Enable User Data Constraint aktiviert und CONFIDENTIAL als Transport Guarantee ausgewählt ist.

- Nun kann das Projekt neu deployed werden. Wenn nun das Projekt SecuredStockQuoter ausgeführt wird, sollte beim Öffnen der WSDL-Datei im Webbrowser das Zertifikat des Services angezeigt werden. Ausserdem ist zu überprüfen ob der Service über das HTTPS-Protokoll aufgerufen wird.

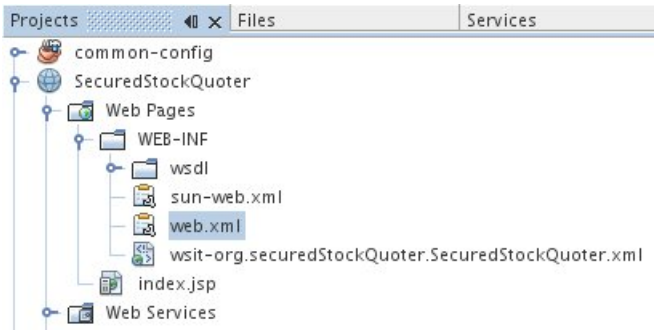

Abbildung C.3: Übersicht web.xml Datei

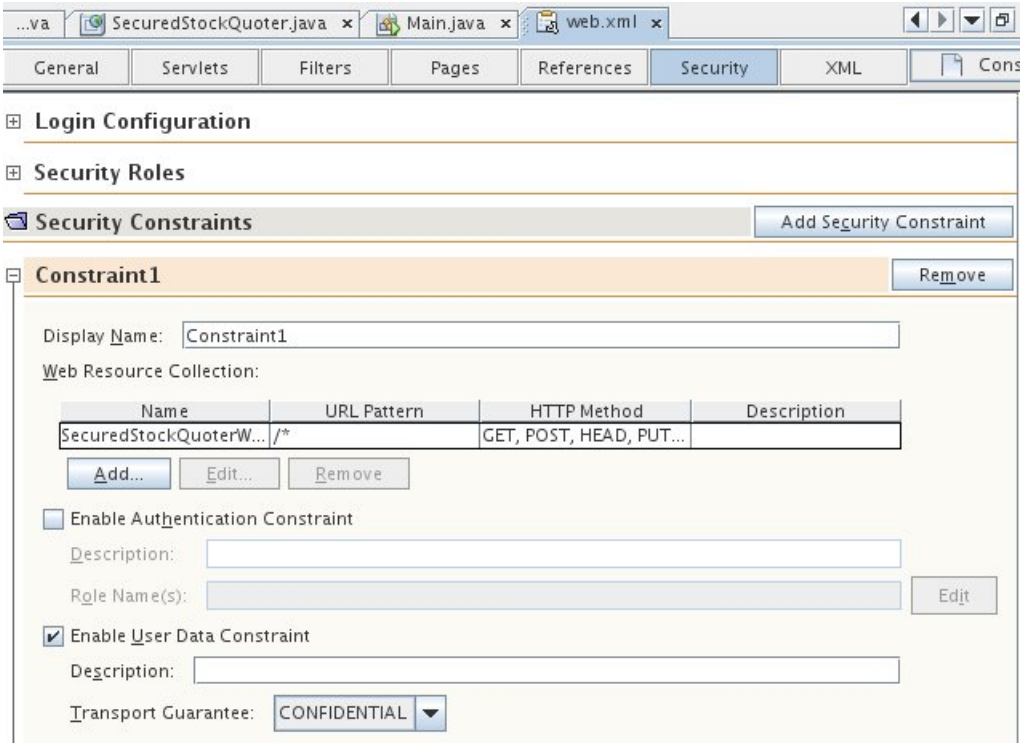

Abbildung C.4: Änderungen web.xml Datei

Die Web Service Referenz im SecuredStockQuoter Client muss nun neu erstellt werden. Beim Erstellen ist zu beachten, dass die URL zur WSDL-Datei des Web Services mit *https* beginnt. Ausserdem muss in der URL der Hostname des Systems, auf welchem der Web Service ausgeführt wird angegeben werden, damit der Client das richtige Zertifikat im Keystore findet.

Da wir für diese Case Study mit den Standard Zertifikaten von NetBeans arbeiten, welche selbstsigniert sind, stuft die Java Virtual Machine diese Zertifikate nicht als vertrauenswürdig ein. Damit das Zertifikat, welches der Web Service verwendet vom Client als vertrauenswürdig angesehen wird, muss in den Eigenschaften des SecuredStockQuoterClient Projekts, wie in Abbildung C.5 dargestellt, unter *run/VM Options* der Keystore, welcher das Zertifikat des Web Services enthält, als Truststore spezifiziert werden. Dies geschieht mit der Option *-Djavax.net.ssl.trustStore*. Ansonsten wird eine *sun.security.provider.certpath.SunCertPathBuilderException* ausgegeben.

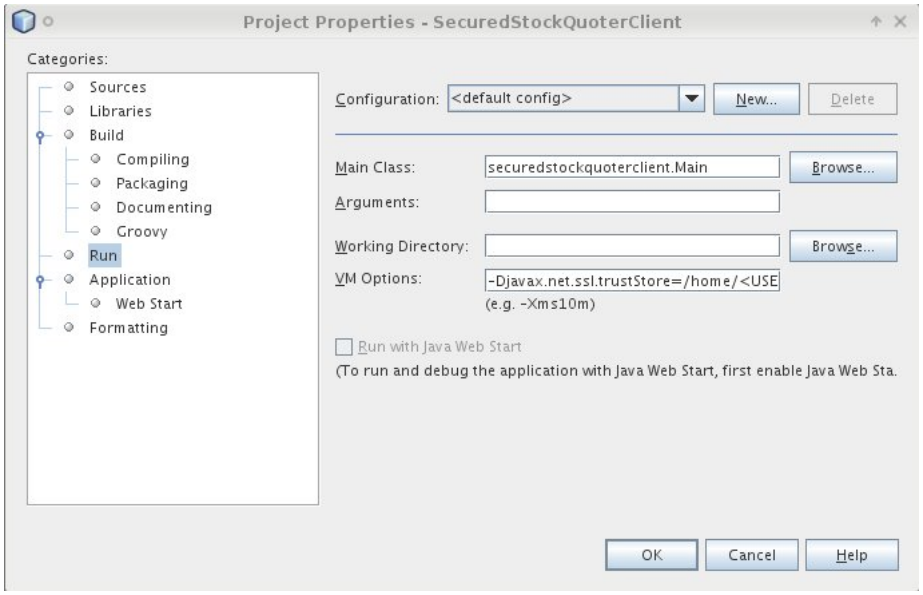

Abbildung C.5: Eigenschaften des SecuredStockQuoterClient Projekts

C.4 Fazit

Die Case Study zeigt, dass sich die Web Services relativ einfach mit TransportSecurity (SSL) sichern lassen. Dadurch, dass NetBeans eine sehr gute Unterstützung für Web Services bietet, wäre der Aufwand einer Implementierung von Kanalsicherheit zwischen den Web Services und den Voter bzw. Tallier Applikationen relativ klein.

Anhang D

Zeitplan

Abbildungeb D.1, D.2 und D.3 geben eine Übersicht, über die zu Beginn des Projekts erstellte Planung.

D Zeitplan

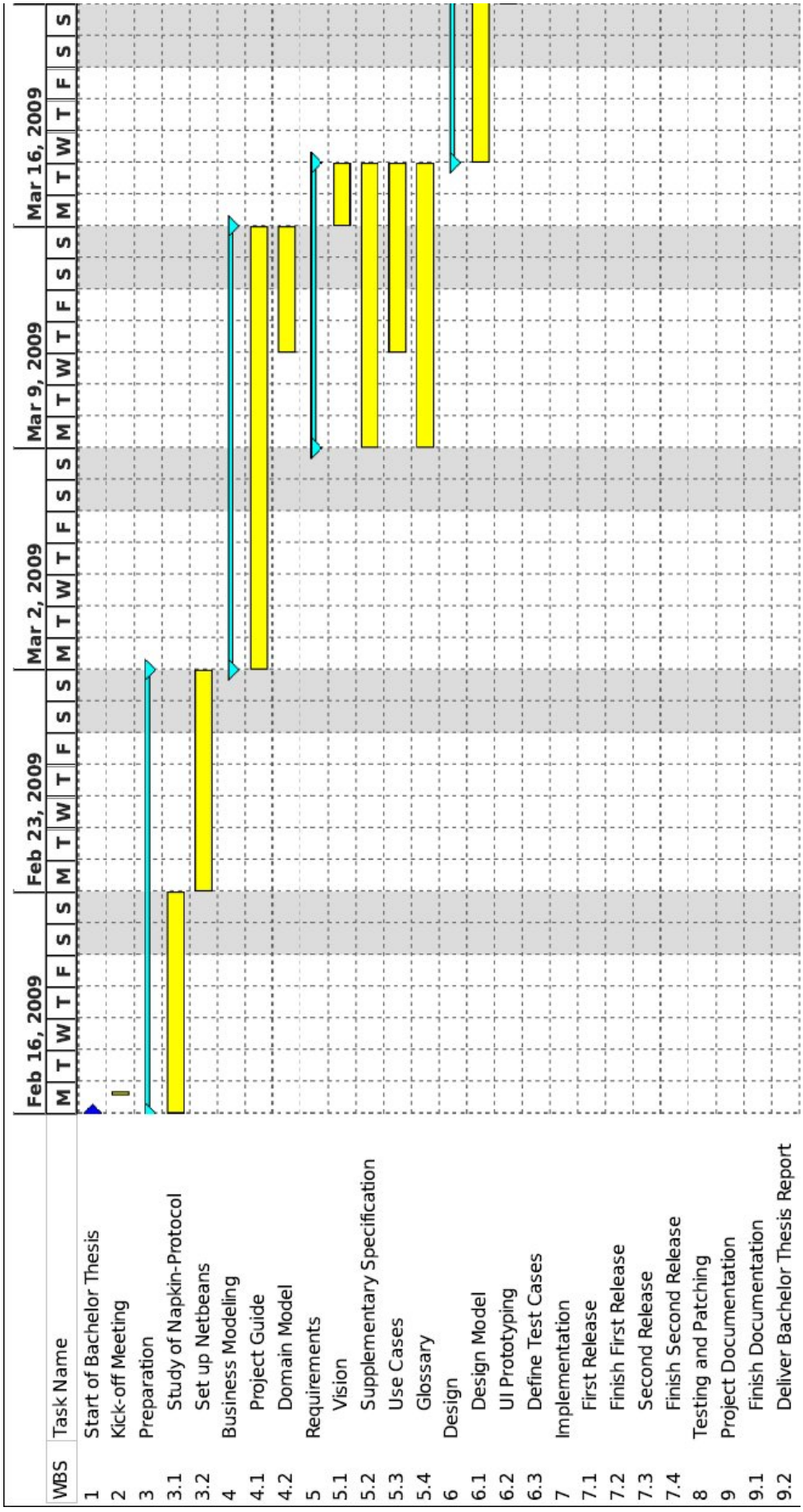

Abbildung D.1: Zeitplan

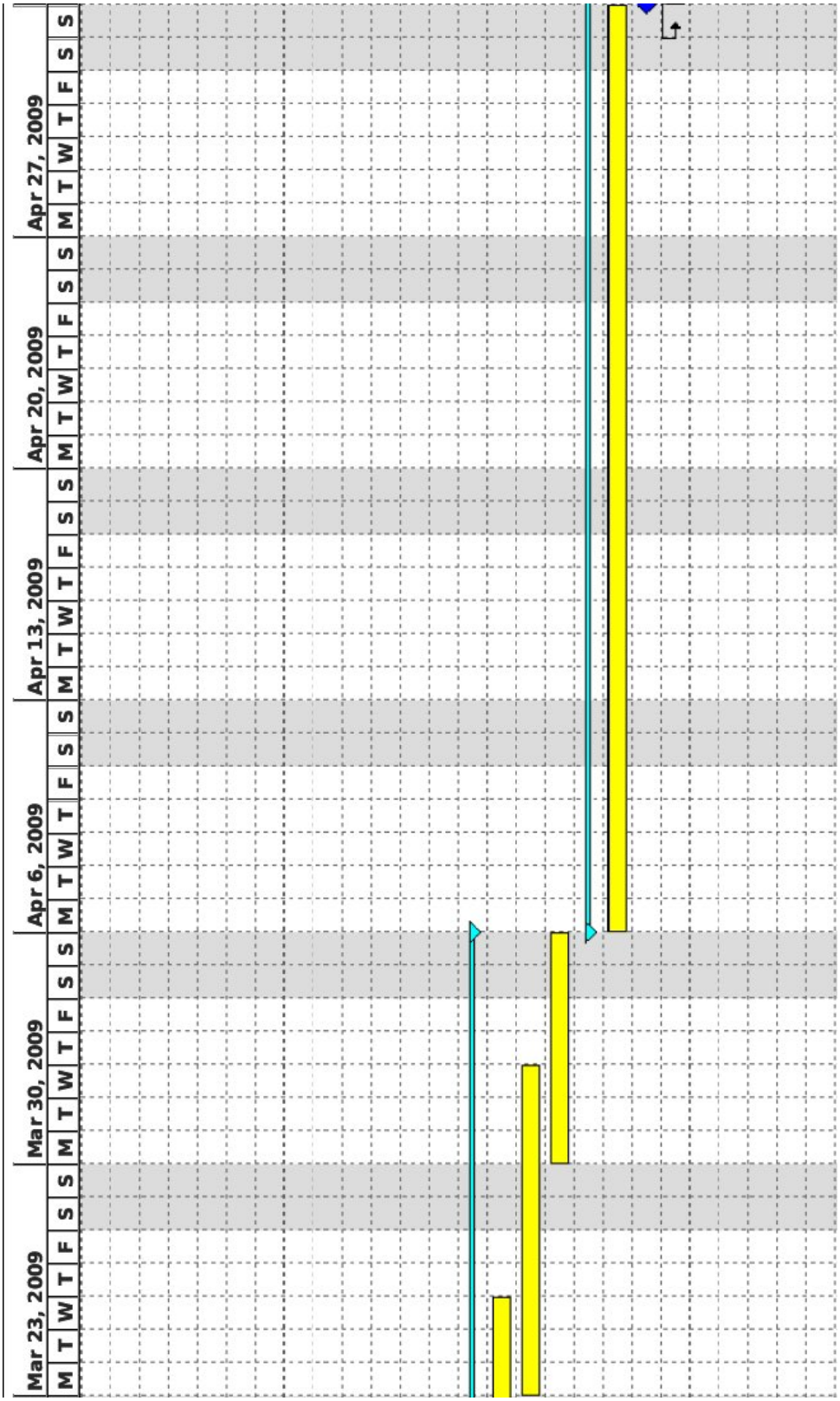

Abbildung D.2: Zeitplan

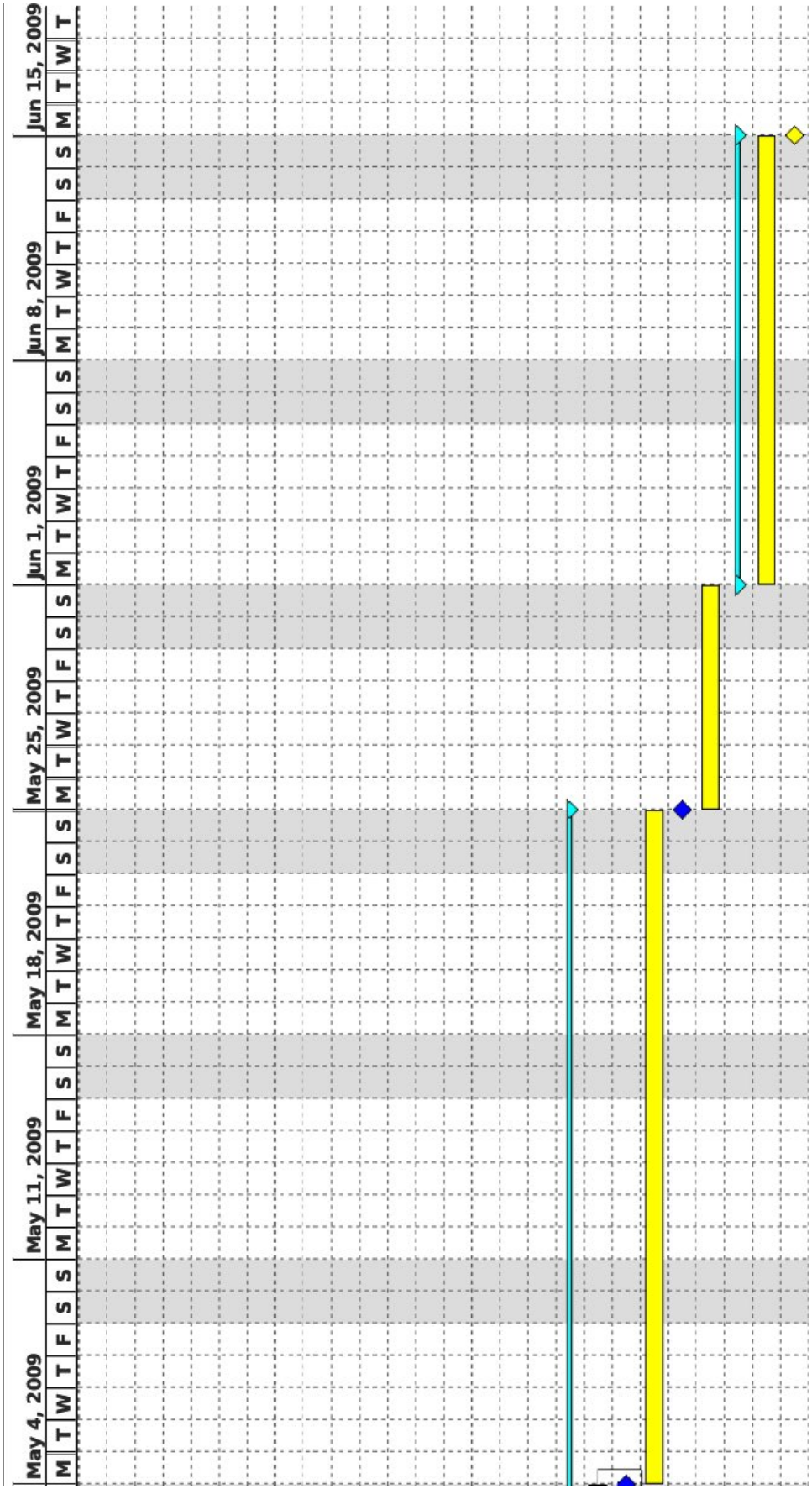

Abbildung D.3: Zeitplan

Literaturverzeichnis

[1] Schweizerische Bundeskanzlei. Pilotprojekte Vote électronique. http://www.bk.admin.ch/themen/pore/evoting/00774/index.html?lang=de, abgerufen am 11. Juni 2009

[2] Haenni, R., König, R., Fischli, S., Hauser, S.: TrustVote: A Hybrid E-Voting System for Large-Scale Elections.

[3] Boldyreva, A.: Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In: Public Key Cryptograhy - PKC 2003. LNCS 2567 (2002) pp 31-46

[4] Shamir, A.: How to share a secret. In: Communications of the ACM 22(11) (1979) pp 612-613

[5] Daemen, J., Rijmen, V.: The Design of Rijndael. AES: The Advanced Encryption Standard (2002)

[6] NESSIE consortium: Portfolio of recommended cryptographic primitives. https://www.cosic.esat.kuleuven.be/nessie/deliverables/decision-final.pdf, abgerufen am 11. Juni 2009

[7] Institute of Standards and Technology (NIST): Secure Hash Standard. http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf, abgerufen am 11. Juni 2009

[8] Wang, X., Yin, Y., Yu, H.: Finding Collisions in the Full SHA-1. In: Crypto'05. Santa Barbara, CA (2005)

[9] Institute of Standards and Technology (NIST): Secure Hashing, Approved Algorithms. http://csrc.nist.gov/groups/ST/toolkit/secure_hashing.html, abgerufen am 11. Juni 2009

[10] Rivest, R., Shamir, A., Adleman, L.: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. In: Communications of the ACM 21,2 (1978) pp 120-126

[11] Chaum, D.: Blind signatures for untraceable payments. In: Advances in Cryptology - Crypto '82. Springer-Verlag (1983) pp 199-203

[12] Niemeyer, P., Knudsen, J.: Learning Java. O'Reilly (2005) pp 851-862

[13] Stallings, W.: Cryptography and network security. Prentice Hall (2006) pp 186-189

[14] RFC 2898: PKCS #5: Password-Based Cryptography Specification Version 2.0 http://www.ietf.org/rfc/rfc2898.txt, abgerufen am 11. Juni 2009

[15] RFC 3369: Cryptographic Message Syntax (CMS) http://www.ietf.org/rfc/rfc3369.txt, pp 26-27, abgerufen am 11. Juni 2009

[16] Larman, C.: Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. Markt und Technik (2005)

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Glossary

A

API Application Programming Interface.

AES Advanced Encryption Standard.

D

DES Data Encryption Standard.

G

GNU GNU is not Unix.

GUI Graphical User Interface.

H

HTTPS HyperText Transfer Protocol Secure.

I

IDE Integrated Development Environment.

J

JAR Java Archive.

JAXB Java Architecture for XML Binding.

JAXWS Java API for XML - Web Services.

JCA Java Cryptography Architecture.

JCE Java Cryptography Extension.

JDK Java Development Kit.

K

KISS Keep It Simple and Smart.

P

PKCS Public Key Cryptography Standards.

S

SHA Secure Hash Algorithm.

SOAP Simple Object Access Protocol.

SSL Secure Socket Layer.

T

TDD Test Driven Development.

U

UI User Interface.

UML Unified Modeling Language.

UP Unified Process.

URL Uniform Resource Locator.

W

WSDL Web Service Description Language.

X

XML Extensible Markup Language.