HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

INSTITUTE FOR
NETWORKED SOLUTIONS

Post-Quantum E-Voting

Studienarbeit

Abteilung Informatik

Hochschule für Technik Rapperswil

Frühjahrsemester 2019

Autoren:	Lukas Lätsch, Rolf Furrer, Romeo Spinas
Betreuer:	Prof. Dr. Andreas Steffen
Projektpartner:	INS, HSR
Abgabedatum:	31.05.2019

1 Aufgabenstellung Semesterarbeit

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

Studienarbeit 2019

Post-Quantum E-Voting

Studenten: Rolf Furrer, Lukas Lätsch, Romeo Spinas

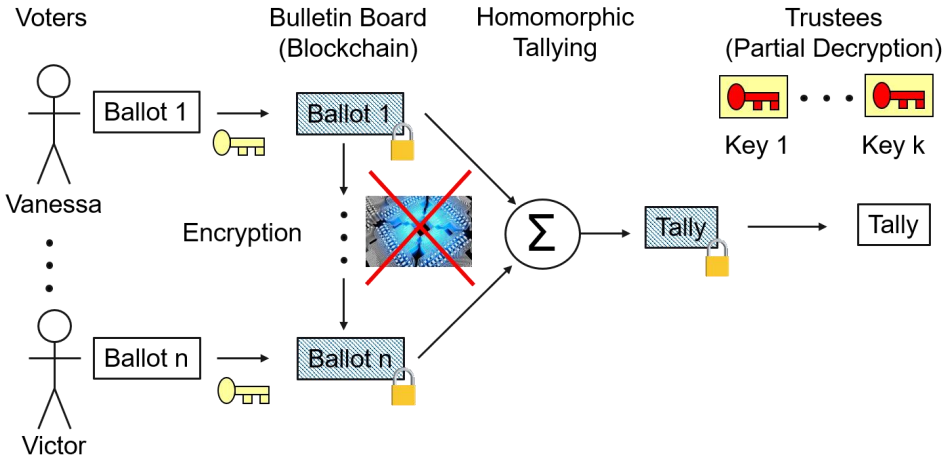
Betreuer: Prof. Dr. Andreas Steffen

Ausgabe: Dienstag, 19. Februar 2019

Abgabe: Freitag, 31. Mai 2019

Einführung

E-Voting über das Internet ist ein aktuelles Thema. Die zur Zeit in der Schweiz und anderen Ländern benutzten Systeme sind nicht sicher, weil die zentralen Server gehackt und damit die Endergebnisse manipuliert werden können. Seit mehr als zehn Jahren gibt es aber sichere Systeme, bei dem jeder Wähler auf einem durch eine Blockchain gesicherten Bulletin-Board jederzeit seine in verschlüsselter Form abgegebene Stimme verifizieren kann und auch die Berechnung des Gesamtresultats aus der Summe aller abgebenen Stimmen überprüfen kann. Dabei wird das Wahlgeheimnis vollständig gewahrt.



Zur Zeit werden diese Ende-zu-Ende verifizierbaren Systeme auf der Basis der RSA und ElGamal Public Key Verfahren realisiert. Es wird erwartet, dass ca. in zehn Jahren diese Algorithmen durch Quantencomputer geknackt werden können. Da die in der Blockchain öffentlich publizierten verschlüsselten Stimmzettel aus Datenschutzgründen mindestens 20-30 Jahre geheim bleiben müssen, sollte man schon jetzt die Realisierung von neuartigen E-Voting Systemen in Angriff nehmen, die resistent gegen Attacken durch Quantencomputer sind.

2016 wurde durch die Doktorandin Ilaria Chillotti ein solches Verfahren [1] beschrieben und die benötigten neuartigen Kryptoalgorithmen in der TFHE Open Source Library [2] zur Verfügung gestellt.

Aufgabenstellung

- Einarbeiten in das LWE-basierte homomorphe E-Voting Verfahren [1, 3].
- Auf der Basis der TFHE Library [2] soll in der ausgeschriebenen Arbeit in C++ ein einfacher Prototyp einer Post-Quantum E-Voting Anwendung erstellt werden.
- Unterstützung einer Ja/Nein Abstimmung und der Wahl eines Vertreters aus N Kandidaten, beide Varianten mit der Möglichkeit der Stimmenthaltung.
- Aufteilung des LWE Schlüssels auf mehrere Trustees.
- Keine Verwendung von Public Key Signaturen, sowie keine Implementation einer Blockchain für das Bulletin Board.
- **Optional:** Mehrere Bulletin-Boards können parallel die homomorphe Aufsummierung der Stimmen vornehmen, um eventuelle Manipulationen aufzudecken.

Links

[1] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène: “An homomorphic LWE based E-voting Scheme”, 2016
https://ilachill.github.io/papers/CGGI16a-An_homomorphic_LWE_based_E-voting_Scheme.pdf

[2] TFHE: Fast Fully Homomorphic Encryption over the Torus. A fast open-source library for fully homomorphic encryption
https://tfhe.github.io/tfhe/

[3] Ilaria Chillotti: “Vers l’efficacité et la sécurité du chiffrement homomorphe et du cloud computing”, 2018
https://ilachill.github.io/papers/these_Ilaria_Chillotti_wo_acknowl.pdf

Rapperswil, 19. Februar 2019

Prof. Dr. Andreas Steffen

2 Abstract

2.1 Einleitung

Die Studienarbeit basiert auf der Doktorarbeit von Ilaria Chillotti[2] zum Thema 'Fast Fully Homomorphic Encryption Library over the Torus'(TFHE). Dabei handelt es sich um eine Verschlüsselung, die es erlaubt, auf verschlüsselten Daten logische Operationen auszuführen. Der Nachteil ist jedoch, dass mehr Performance nötig ist um die Berechnungen durchzuführen. Die Arbeit von Ilaria Chillotti und ihrer Forschungskollegen haben die Geschwindigkeit der Verschlüsselung verbessert. Dadurch ist es möglich diese Verschlüsselungsart ohne immensen Zeitaufwand einzusetzen. Bei E-Voting Systemen mit konventionellen Verschlüsselungen werden die Stimmen vor dem Zusammenzählen entschlüsselt. Die homomorphe Verschlüsselung ermöglicht es die verschlüsselten Stimmen zu zählen und somit das Wahlgeheimnis zu bewahren. Ein weiteres Schwäche einiger Verschlüsselung ist das Entwicklungsstand der Quantencomputer. Schätzungsweise wird intern 10 Jahren ein Grossteil der heutigen Verschlüsselungen unsicher, das ist bei TFHE nicht der Fall.

2.2 Aufgabenstellung

Im Rahmen der Arbeit mussten wir auf der Basis der TFHE Library in C++ einen einfachen Prototypen einer Post-Quantum E-Voting Anwendung erstellen. Der Prototyp muss Ja/Nein Abstimmung und der Wahl eines Vertreters aus mehreren Kandidaten unterstützen. Zusätzlich muss auch eine Stimmenthaltung ermöglicht werden.Zur Verhinderung von Wahlbetrug mussten zudem die Schlüssel auf unabhängige Parteien verteilt werden können. Eine zusätzliche optionale Anforderung ist der Einsatz mehrerer Stimmenzähler, um eine weitere Betrugsmöglichkeit eliminieren zu können. Im Arbeitsumfang ist die Verifikation der Wähler durch die Verwendung von Public Key Signaturen, sowie die Sicherstellung der Nachvollziehbarkeit durch den Einsatz einer Blockchain nicht enthalten.

2.3 Ergebnisse

Das wichtigste Ergebnis unserer Arbeit ist die Implementation eines E-Voting Prototypen. Zu diesem Zweck haben wir in ersten Phase die bestehende Library überarbeitet. Dies beinhaltet den Einsatz von Objektorientierung sowie das anheben des Standards auf C++17. Zudem haben wir den Funktionsumfang der Library erweitert. Beispielsweise braucht es für das E-Voting eine Funktion zum Zusammenzählen der Stimmen. Des Weiteren haben wir bei der Überarbeitung sicherheitstechnische Aspekte beachtet.
In einer zweiten Phase haben wir einen Prototypen mit vier Teilapplikationen erstellt. Es handelt sich dabei um den Coordinator, Trustee, das Bulletin Board und den Voter. Eine Abstimmungsdurchführung kann vereinfacht mit einem Coordinator, Trustee, Bulletin Board und einem bis mehreren Voter durchgeführt werden.
Die Überarbeitung der Library und die Modularisierung des Prototypen hat dazu geführt, dass der neue erstellte Code deutlich lesbarer geworden ist, zusätzlich ist die Anwendung übersichtlicher und kürzer geworden.
Nach Performance Tests haben wir eine geringfügige Differenz der Ausführungszeit festgestellt. Das liegt vorwiegend an der Objektorientierung und der Löschung der sensiblen Daten.

Inhaltsverzeichnis

Kapitel 1: Aufgabenstellung In diesem Abschnitt wird die offizielle Aufgabenstellung vom betreuenden Dozenten aufgeführt.

Kapitel 2: Abstract Im Abstract Kapitel ist die Arbeit zusammengefasst.

Kapitel 3: Einleitung Die Einleitung beschreibt die TFHE näher. Sie zeigt auch auf die Verschlüsselungsalgorithmen für eine symmetrische Verschlüsselung verwendet werden kann. Des Weiteren wird gezeigt wie das symmetrische und das public/private Verfahren mit TFHE funktioniert. Es werden die Gate Funktionen näher beschrieben.

Kapitel 4: Design Das Kapitel zeigt den Aufbau der Applikation auf. Es werden die verschiedenen Komponenten näher beschrieben und auf die Anforderungen eingegangen.

Kapitel 5: Ergebnisse Die wichtigsten Ergebnisse der Arbeit werden in diesem Kapitel detailiert beschrieben.

Kapitel 6: Schlussfolgerung In der Schlussfolgerung wird der Status der Arbeit analysiert.

Kapitel 7: Projekt Management Im Projekt Management wird auf welches die anspruchsvollen Teile der Entwicklung des E-Voting-Systems sind.

Kapitel 8: Abkürzungsverzeichnis In diesem Anhang werden die verwendeten Abkürzungen aufgelistet und ausgeschrieben.

Appendix A: Reflexion In diesem Appendix werden die selbstkritischen Reflexionen der Teammitglieder und deren Erfahrungen aufgeführt.

Appendix B: Eigenständigkeitserklärung Der Appendix enthält die Eigenständigkeitserklärungen der Teammitglieder.

Appendix C: Projektplan Der Projektplan der zu Beginn der Arbeit festgelegt wurde ist hier beigelegt.

Appendix D: Anforderungsspezifikation Die Anforderungsspezifikation sind hier angehängt, welche zu Beginn der Arbeit erstellt worden ist.

Appendix E: Software Architektur Die Software Architektur beschreibt die Architektur des Software Projektes, welche nach den Anforderungsspezifikation erstellt worden ist.

Appendix F: Software-Lizenz Die Nutzungsrechte der Arbeit und Software sind hier aufgeführt.

Appendix G: Bedienungsanleitung Software In diesem Kapitel ist beschrieben, wie die Applikation kompiliert werden kann und wie man sie bedient.

3 Einleitung

Im Rahmen unserer Studienarbeit haben wir eine Grundlage für ein E-Votingsystem basierend auf TFHE (Fast Fully Homomorphic Encryption over the Torus) implementiert. Im folgenden Kapitel werden wir auf den Aufbau von TFHE genauer eingehen.

3.1 Homomorphe Verschlüsselung

Homomorphe Verschlüsselung ist eine Art von Verschlüsselung, die es erlaubt mathematische oder logische Operationen auf den verschlüsselten Daten auszuführen. Das Resultat der verschlüsselten Operation entspricht dabei dem Resultat der Operation auf den nicht verschlüsselten Daten. Diese Art von Verschlüsslung wird oft zum Schutz der Privatsphäre oder von Firmengeheimnissen verwendet. Beispielsweise können so Statistiken über eine Gruppe erhoben werden, ohne dass ein Individuum persönliche Informationen preisgeben muss.

3.2 Post-Quanten-Kryptographie

Die schnell fortschreitende Entwicklung der Quantencomputer setzt die Kryptowelt unter Druck. Einige der meist verbreiteten Verschlüsselungen drohen unsicher zu werden. Die Lösung sind Post-Quantum Verschlüsselungen. Eine Post-Quantum-Verschlüsselung beschreibt einen Verschlüsselungsalgorithmus, der den Quantencomputern standhält. LWE erreicht diese Eigenschaft durch das addieren von Rauschen. Das Rauschen erschwert das Faktorisieren stark, da es keine exakte Lösung mehr geben kann. Das Rauschen erschwert allerdings die Homomorphen Operationen, da es stärker wird durch jede Operation. Daraus resultiert eine Abwägung zwischen einem zu hohen Rauschpegel der Homomorphe Operationen verunmöglicht und einem zu tiefen der die Sicherheit reduziert.

3.3 Eigenschaften von TFHE

TFHE erlaubt es, logische Operationen bitweise auf verschlüsselten Daten durchzuführen. Diese Eigenschaft ermöglicht es theoretisch, beliebige binäre Schaltungen nachzubilden. So ist es möglich Daten zu verschlüsseln und sie einem nicht vertrauenswürdigen Computer zu geben, um eine beliebige Rechenoperation auf den Daten ausführen zu können. Der Rechnende Computer ist dabei nicht in der Lage, die Daten zu im Klartext zu sehen, mit denen er rechnet. TFHE kann grundsätzlich als symmetrische Verschlüsselung eingesetzt werden. Mit einem LWE Schlüssel können Daten verschlüsselt und entschlüsselt werden. Allerdings wäre bei einer Schlüssellänge von 500 Elementen ein Ciphertext von 16032bit pro Datenbit nötig. Public/Private Key Verschlüsselung ist auch möglich.

3.3.1 Symmetrische Verschlüsslung und Entschlüsselung

Definieren von Parametern und Schlüsselgenerierung
Als zu definierende Parameter gibt es auf der symmetrische Verschlüsslung und Entschlüsselung lediglich die Schlüssellänge. Im folgenden Beispiel ist die Schlüssellänge 8 und sieht wie folgt aus.

LWE Key						Länge	8
1	0	0	1	0	0	1	1

Tabelle 1: Symmetrischer LWE Schlüssel

Verschlüsslung von Daten
Zur Verschlüsselung müssen zunächst die Daten in ein LWE Sample transformiert werden. Ein LWE Sample besteht aus einer Liste (Tabelle 2) von der Grösse der Schlüssellänge an zufällig gewählten Zahlen von der Grösse des minimalen Int bis zum maximalen Int. Für das Beispiel von -4/8 bis 4/8.

LWE Sample									Länge	8
A	-2/8	4/8	-1/8	-3/8	4/8	-3/8	1/8	-3/8	**B**	

Tabelle 2: LWE Sample mit zufällig gewählten A-Koeffizienten

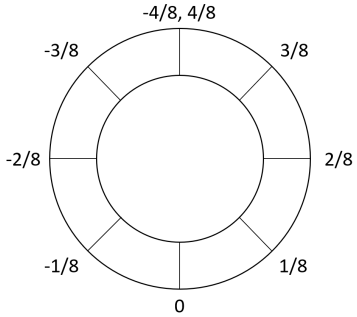


Abbildung 1: Torus Mapping

Zusätzlich gibt es ein Element B (Tabelle 2), das die Daten enthält. Dafür gibt es die folgende Zuweisung: eine logische Null (false) wird zu -1/8 und logisch Eins (true) wird zu 1/8. Angenommen, wir wollten 0b10 verschlüsseln, hätten wir beispielsweise die folgenden LWE Samples:

LWE Sample									Länge	8
A	-2/8	4/8	-1/8	-3/8	4/8	-3/8	1/8	-3/8	**B**	1/8

Tabelle 3: Erstes LWE Sample mit zufällig gewählten A-Koeffizienten und gesetztem B Element

LWE Sample									Länge	8
A	-4/8	1/8	-2/8	3/8	-3/8	-1/8	-2/8	4/8	**B**	-1/8

Tabelle 4: Zweites LWE Sample mit zufällig gewählten A-Koeffizienten und gesetztem B Element

Nun wird noch das B Element verschlüsselt. Hierfür werden die A Komponenten mit dem Schlüssel multipliziert und zur B Komponente addiert, was folgend illustriert wird.

$$B' = B + \begin{bmatrix} K_1 & K_2 & K_3 & K_4 & K_5 & K_6 & K_7 & K_8 \end{bmatrix} * \begin{bmatrix} A_1 \\ A_2 \\ A_3 \\ A_4 \\ A_5 \\ A_6 \\ A_7 \\ A_8 \end{bmatrix}$$

In diesem Beispiel gäbe das die folgenden Samples:

LWE Sample									Länge	8
A	-2/8	4/8	-1/8	-3/8	4/8	-3/8	1/8	-3/8	**B**	2/8

*Tabelle 5: Erstes **verschlüsseltes** LWE Sample*

LWE Sample									Länge	8
A	-4/8	1/8	-2/8	3/8	-3/8	-1/8	-2/8	4/8	**B**	0/8

*Tabelle 6: Zweites **verschlüsseltes** LWE Sample*

Entschlüsselung der Daten
Um die Daten wieder entschlüsseln zu können, muss nun die A Komponente wieder von der B Komponente abgezogen werden.

$$B = B' - \begin{bmatrix} K_1 & K_2 & K_3 & K_4 & K_5 & K_6 & K_7 & K_8 \end{bmatrix} * \begin{bmatrix} A_1 \\ A_2 \\ A_3 \\ A_4 \\ A_5 \\ A_6 \\ A_7 \\ A_8 \end{bmatrix}$$

So kommt man wieder zu den initialen LWE Samples.

LWE Sample									Länge	8
A	-2/8	4/8	-1/8	-3/8	4/8	-3/8	1/8	-3/8	**B**	1/8

Tabelle 7: Erstes initiale LWE Sample

LWE Sample									Länge	8
A	-4/8	1/8	-2/8	3/8	-3/8	-1/8	-2/8	4/8	**B**	-1/8

Tabelle 8: Zweites initiale LWE Sample

Zum Schluss muss man nur noch entscheiden, ob es sich um eine logische Eins oder Null handelt. Dafür kann man nach $B > 0$ vergleichen, wobei $B > 0 \iff Eins$ und $B \leqslant 0 \iff Null$ entspricht.

3.3.2 Public/Private Key Verschlüsslung

Definieren von Parametern und Schlüsselgenerierung

Als zu definierende Parameter gibt es für die Public/Private Key Verschlüsslung lediglich die Schlüssellänge. Im folgenden Beispiel sind es 8. Der private Schlüssel hat dieselbe Struktur wie derjenige der symmetrischen Verschlüsslung.

LWE Key						Länge	8
1	0	0	1	0	0	1	1

Tabelle 9: Asymetrischer LWE Schlüssel

Zusätzlich müssen nun aber noch öffentliche Schlüssel generiert werden. Diese öffentlichen Schlüssel können erstellt werden, indem die Zahl Null symmetrisch verschlüsselt wird.

LWE Sample									Länge	8
A	-2/8	4/8	-1/8	-3/8	4/8	-3/8	1/8	-3/8	**B**	1/8

Tabelle 10: Erstes LWE Sample mit zufällig gewählten A-Koeffizienten und gesetztem B Element

LWE Sample									Länge	8
A	-4/8	1/8	-2/8	3/8	-3/8	-2/8	-2/8	1/8	**B**	-2/8

Tabelle 11: Erstes LWE Sample mit zufällig gewählten A-Koeffizienten und gesetztem B Element

Verschlüsslung von Daten

Zur Verschlüsselung müssen zunächst die Daten in ein LWE Sample transformiert werden. Ähnlich der symmetrischen Verschlüsselung jedoch benötigt man ein triviales LWE Sample. Das bedeutet, dass die A Komponente aus Nullen besteht und lediglich die B Komponente zugewiesen werden muss.
Für den Wert 0b10 ergeben sich so die folgenden Samples:

LWE Sample									Länge	8
A	0	0	0	0	0	0	0	0	**B**	1/8

Tabelle 12: Erstes triviales LWE Sample

LWE Sample									Länge	8
A	0	0	0	0	0	0	0	0	**B**	-1/8

Tabelle 13: Zweites triviales LWE Sample

Anschliessend können ein oder mehrere der öffentliche Schlüssel (OS) zum Sample (S) addiert werden. Dabei gilt $S'_{A[i]} = S_{A[i]} + OS_{A[i]}$ für $i = 0$ bis $i = Schlüssellänge$ und $S'_B = S_B + OS_B$.

LWE Sample									Länge	8
A	-2/8	4/8	-1/8	-3/8	4/8	-3/8	1/8	-3/8	**B**	2/8

*Tabelle 14: Erstes **verschlüsseltes** LWE Sample*

LWE Sample									Länge	8
A	-4/8	1/8	-2/8	3/8	-3/8	-2/8	-2/8	1/8	**B**	-3/8

*Tabelle 15: Zweites **verschlüsseltes** LWE Sample*

Entschlüsselung der Daten
Zur Entschlüsselung der Daten kann nun die symmetrische Entschlüsselung im Kapitel 3.3.1 angewendet werden.

3.3.3 Gatterfunktionen

Die Gatterfunktionen erlauben es, logische Operationen auf den verschlüsselten Daten anzuwenden. So war es uns beispielsweise möglich, die Stimmen zu zählen, ohne eine einzelne Stimme kennen zu müssen. TFHE unterstützt die Ausführung der folgenden 12 Operationen: and, andyn, andny, nand, or, oryn, orny, nor, xor, not und mux. Damit kann TFHE grundsätzlich eine beliebig komplexe Schaltung abbilden, jedoch kann der Zeitaufwand einer logischen Operation zum Problem werden. Das Aufaddieren einer einzelnen Stimme mit 8 Varianten dauert beispielsweise etwa 5 Sekunden. Die Ursache ist, dass bei der Verschlüsselung ein Rauschen auf alle Koeffizienten addiert wird, um eine Faktorisierung zu erschweren. Dieses Rauschen vergrössert sich mit jeder Operation. Ab einem gewissen Rauschpegel ist es möglich, Rechenfehler zu machen. Um dies zu verhindern, kann mittels einem Keyswitch das Rauschen entfernt werden, jedoch ist diese Operation sehr aufwändig. Eine mögliche Optimierung ist es, möglichst viele Operationen ohne einen Keyswitch auszuführen. Aus Zeitgründen haben wir diese Optimierung für Leveled LHE jedoch nicht implementiert.

3.3.4 Beispiel einer OR Operation

Verschlüsslung von Daten
Die Verwendung von Gatterfunktionen erfordert die Verwendung der gesamten Library. Die Generierung dieser Parameter ist deutlich aufwändiger als die Generierung der symmetrischen Verschlüsselung, deshalb wird sie in diesem Beispiel übersprungen. Es gilt aber wieder die Schlüssellänge des LWE Schlüssels von 8. Zusätzlich kann derselbe LWE Schlüssel verwendet werden.

LWE Key						Länge	8
1	0	0	1	0	0	1	1

Tabelle 16: LWE Key

Zusätzlich benötigt man 2 LWE Samples die mittels symmetrischer, asymmetrischer Verschlüsselung oder durch vorhergehende Gatteroperationen entstanden sind:

LWE Sample									Länge	8	Rauschen
A	-2/8	4/8	-1/8	-3/8	4/8	-3/8	1/8	-3/8	**B**	2/8	n_1

Tabelle 17: Erstes LWE Sample

LWE Sample									Länge	8	Rauschen
A	-4/8	1/8	-2/8	3/8	-3/8	-1/8	-2/8	4/8	**B**	-1/8	n_2

Tabelle 18: Zweites LWE Sample

Logische Operation
Für die OR-Operation wird zuerst eine Konstante benötigt.

LWE Sample									Länge	8	Rauschen
A	0	0	0	0	0	0	0	0	**B**	1/8	0

Tabelle 19: Triviales LWE Sample

Als zweites werden die beiden Samples und die OR-Konstante im Fall der OR-Operation aufaddiert. Die Konstante und die Rechenoperationen variieren abhängig der logischen Operation. NAND beispielsweise benötigt dieselbe Konstante, verwendet jedoch Subtraktionen.

LWE Sample									Länge	8	Rauschen
A	0	0	0	0	0	0	0	0	**B**	1/8	0
A	-2/8	4/8	-1/8	-3/8	4/8	-3/8	1/8	-3/8	**B**	2/8	n_1
A	-4/8	1/8	-2/8	3/8	-3/8	-2/8	-2/8	1/8	**B**	-3/8	n_2
A	2/8	-3/8	-3/8	0	1/8	3/8	-1/8	-2/8	**B**	1/8	$n_1 + n_2$

Tabelle 20: OR Operation

Anschliessend wird der Keyswitch ausgeführt. Auf Grund der Komplexität wird dieser Schritt in diesem Beispiel übersprungen.

Entschlüsselung der Daten
Zur Entschlüsselung der Daten kann nun die symmetrische Entschlüsselung angewendet werden.

3.3.5 Schlüssel

TFHE kennt drei Schlüsseltypen und zwei Schlüsselcontainer, die für die Gatteroperationen benötigt werden. Sie sind hierarchisch aufgebaut und dienen verschiedenen Zwecken.

LWE Schlüssel
Der LWE Schlüssel dient zur Ver- und Entschlüsselung der Daten.

TLWE Schlüssel
Der TLWE Schlüssel ist ein Schlüssel, der hauptsächlich für die Generierung des KeySwitchKey benötigt wird. Er dient selbst nicht direkt der Verschlüsslung von Daten.

TGSW Schlüssel
Wie der TLWE Schlüssel dient der TGSW Schlüssel der Generierung des KeySwitchKey. Zusätzlich ist er lediglich ein Container für den TLWE Schlüssel.

KeySwitchKey
Der KeySwitchKey ist neben dem LWE der wichtigste Schlüssel. Der KeySwitchKey kann einerseits einen LWE Schlüssel durch einen anderen LWE Schlüssel ersetzen, ohne die Daten entschlüsseln zu müssen. Andererseits wird er benötigt, um das Rauschen, das durch die Gatteroperationen verstärkt wird, zu filtern.

LweBootstrappingKey
Der BootstrappingKey ist ein Container für den öffentlichen TGSW Schlüssel. Er wird hauptsächlich zur Generierung des KeySwitchKey benötigt.

3.3.6 Anwendung der Schlüssel für E-Voting

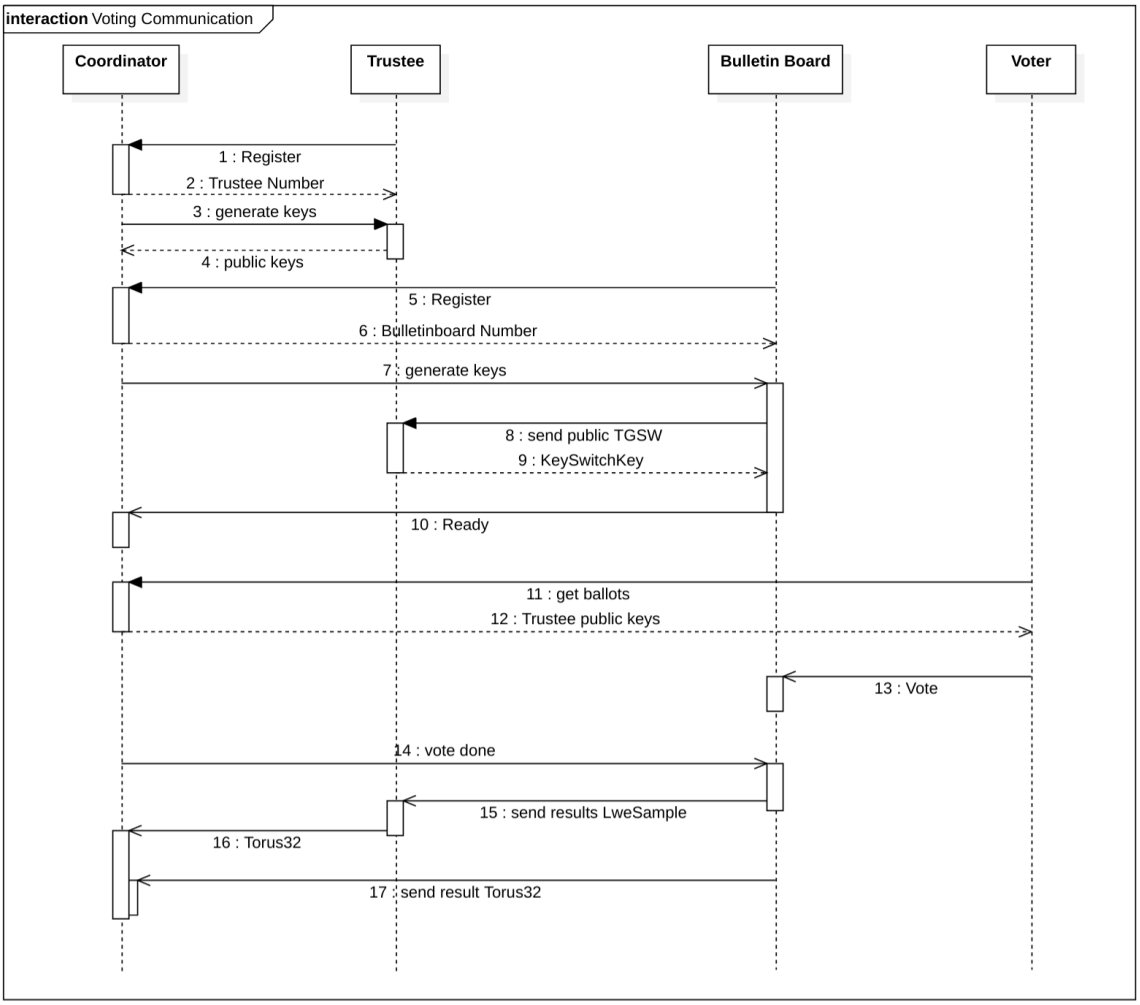


Abbildung 2: Anwendung der Schlüssel für E-Voting

Vor der Wahl müssen sich alle Beteiligten einigen über die verwendeten Schüssellängen. Eine oder mehrere vertrauenswürdige Quellen müssen anschliessend auf Grund der definierten Parameter private und öffentliche LWE Schlüssel generieren. Unabhängig davon müssen ein oder mehrere Stimmenzähler ernannt werden. Die Stimmenzähler generieren anschliessend private und öffentliche TGWS Schlüssel. Die öffentlichen Schlüssel müssen sie im Anschluss alle vertrauenswürdigen Quelle senden, um sie mit dem LWE Schlüssel überschlüsseln zu lassen. In Anschluss können die Stimmenzähler die KeySwitchKeys generieren.
Nun kann die Abstimmung beginnen. Die Wähler können von den vertrauenswürdigen Quellen die öffentlichen LWE Schlüssel beziehen, um ihre Wahlzettel zu verschlüsseln. Anschliessend können sie den Wahlzettel an alle Stimmenzähler senden. Die Stimmenzähler können während der Wahl beginnen, die Stimmen zu zählen, da sie das Ergebnis nicht entschlüsseln können. Wenn die Wahl abgeschlossen ist, senden alle Stimmenzähler ihre Resultate an die vertrauenswürdigen Quellen. Diese können dann je eine Teilentschlüsselung der Resultate vornehmen, um anschliessend gemeinsam das gesamte Resultat zu erhalten.

4 Architektur und Design

4.1 Übersicht

Für die Umsetzung des E-Voting-Systems haben wir vier Komponenten vorgesehen. Einerseits gibt es die vier Komponenten Coordinator, Voter, Trustee und Bulletinboard, die auch unterschiedliche Teilapplikationen darstellen. Zusätzlich gibt es die Core-Library, welche die gemeinsamen Nenner aller Teilapplikationen implementiert.

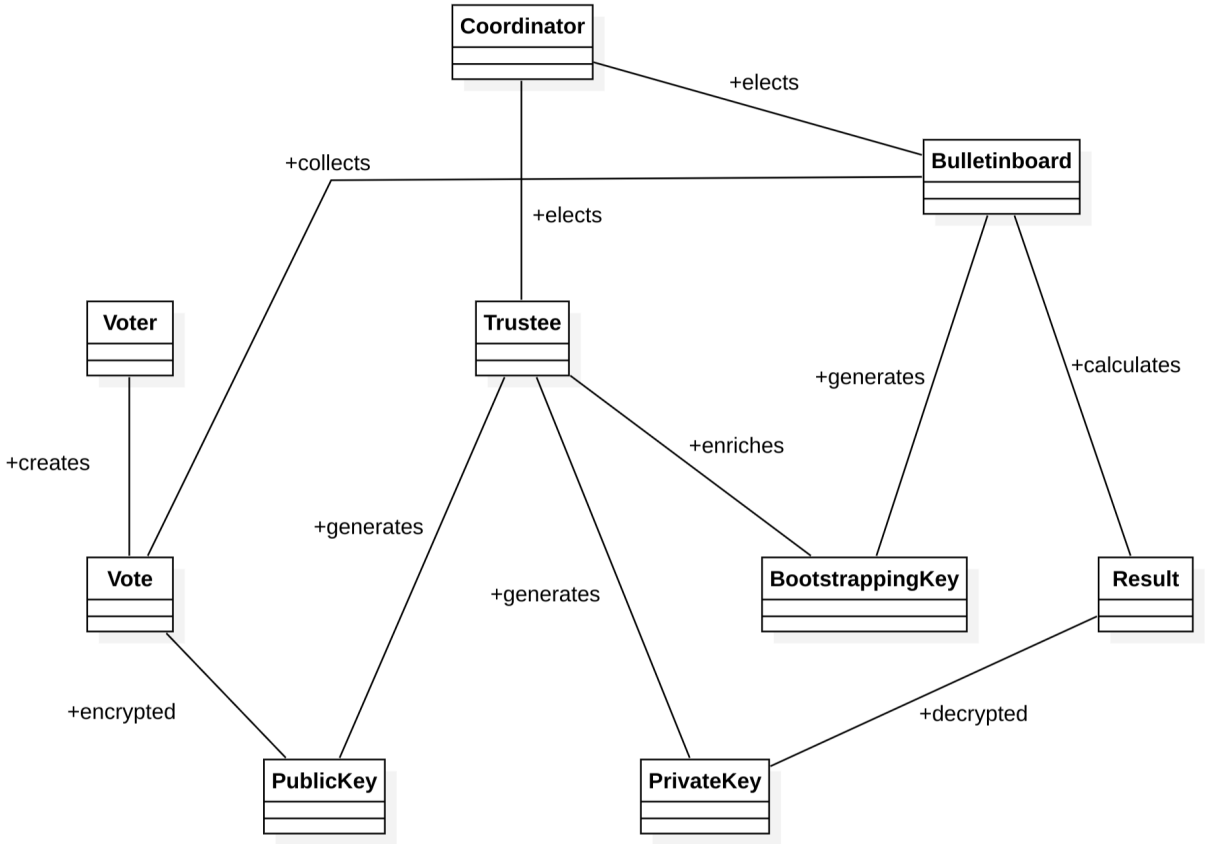


Abbildung 3: Domainmodel

4.1.1 Core-Library

Die Core-Library basiert auf dem TFHE-Projekt auf GitHub [1]. Die originale Codebasis ist in C/C++, meist C++11 konform. Diese bestehende Library erschwert eine saubere Softwarearchitektur, da die Funktionen eine hohe Kopplung aufweisen, über mehrere Dateien verstreut sind und eine Kapselung durch Klassen nicht besteht. Aus diesen Gründen haben wir uns dazu entschieden, die Library mit Hilfe von Klassen zu strukturieren und gleichzeitig auf den C++17-Standard zu portieren. Für die Schnittstellen zu den anderen Programmteilen haben wir uns ebenfalls für C++ entschieden. Da wir für alle Programmteile C++ verwenden, kann der Aufwand für die Kommunikation zwischen den Programmteilen klein gehalten werden. Eine Erweiterung für andere Sprachen mit einer C Schnittstelle, kann zu einem späteren Zeitpunkt nachgerüstet werden.

4.1.2 Trustee

Die Trustees sind für die Schlüsselgenerierung verantwortlich. Zudem halten sie sich bereit, den Votern die öffentlichen Schlüssel zur Verfügung zu stellen. Am Ende der Abstimmung entschlüsselt zudem jeder Trustee seinen Teil der Verschlüsselung. Ebenso generieren die Trustees den KeySwitchingKey für das Bulletin Board.

[1] https://github.com/tfhe/tfhe

4.1.3 Bulletin Board

Das Bulletin Board sammelt die Stimmen und addiert diese auf homomorph verschlüsselt auf.

4.1.4 Voter

Die Voter Komponente ist die Benutzeroberfläche für die Wähler. Der Voter bekommt die Wahlzettel vom Coordinator. Er füllt den Wahlzettel aus und leitet sie an die Bulletin Boards weiter.

4.1.5 Coordinator

Der Coordinator organisiert die Interaktion zwischen den verschiedenen Systemen. Der Trustees und Bulletin Boards melden sich beim Coordinator. Danach werden die Aufträge von ihm verteilt. Der Coordinator ist nicht immer die Zwischenstelle, sodass er seine Position nicht missbrauchen kann.

4.2 Funktionale Anforderungen

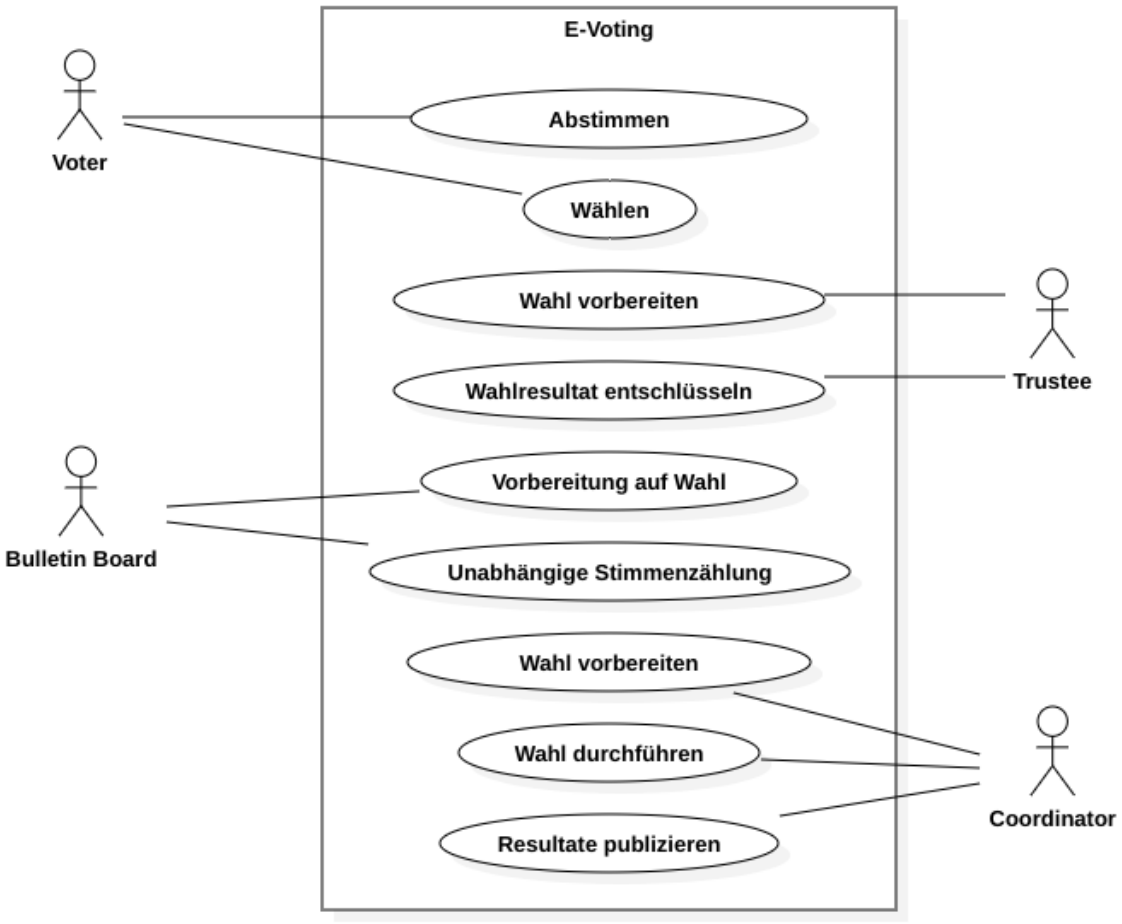


Abbildung 4: Usecases

Die Ausführlichen Usecases sind im Anhang D im Kapitel 1 zu finden.

4.3 Nicht Funktionale Anforderungen

Die nicht Funktionalen Anforderungen sind im Anhang D im Kapitel 5 zu finden.

5 Ergebnisse

5.1 Migrierung von alter Library in C zu neuer in C++

Die Anpassungen der Library auf C++17 sind **fast** abgeschlossen. Die Funktionen sind zusammengefasst in Klassen.

5.2 Gates Algorithm

See picture meeting explained

5.3 Performance Tests

Um die Performance der neuen Applikation zu testen und so einen Vergleich mit der Alten zu ermöglichen, haben wir dazu Tests geschreiben. Die Single Core Geschwindigkeit ist immernoch sehr langsam z.B bei 8 Wahlmöglichkeiten werden pro Stimme im Schnitt 5 Sekunden gebraucht. Das beinhaltet das Dazuzählen einer Stimme und das Entschlüsseln einer Stimme. Die alte Version benötigt ebenfalls etwa 5 Sekunden pro Stimme.

	Version mit alter Library	Version mit neuer Library
4 Varianten	3.3s	1.7s
8 Varinaten	5s	5s

Tabelle 21: Performance Vergleich

Es ist möglich eine beliebige Anzahl von Auswahlmöglichkeiten bereitzustellen, das erhöht den Rechenaufwand jedoch beträchtlich. Bedingt durch die Rechenoperationen auf homomorph verschlüsselten Daten, müssen die Operationen nämlich auf jede Auswahlmöglichkeit angewendet werden, im Grunde werden dort mit dem gleichen Verfahren aber nur Nullen dazugezählt.

Anzahl Varianten	4	8	16	32	64	128
Zeit in s	1.7	5	13.7	34.8	83.3	193.6

Tabelle 22: Performance skalierung

5.4 Operationen auf verschlüsselten Daten

5.4.1 Entwicklung

FullAdder
Die abgegebenen Stimmen auf den Bulletin Boards werden dort individuell aufsummiert. Diese Summierung wird auf verschlüsselten Daten durchgeführt, was möglich ist dank der homomorphen Verschlüsselung. Die Aufsummierung wird in der TFheGateBootstrappingCloudKeySet Klasse mit der fullAdder() Funktion durchgeführt.
Die Schwierigkeit lag darin, dass mit Debuggen die Transformation der Daten nicht nachvollzogen werden konnten, da sie verschlüsselt sind. Die Verifizierung kann nur beim Vergleich zwischen unverschlüsselten Daten (input) und dann den entschlüsselten Daten (output) überprüft werden.

FullSubtractor
In den Abuse Cases wurde auch definiert, dass eine einzelne Stimmabgabe rückgängig gemacht werden sollte, wenn zum Beispiel die Person beim Abstimmen beeinflusst wurde. Somit kann die abstimmende Person vor Abstimmungsende ihre Stimme zurückziehen und neu abstimmen. Da die Stimmabgaben kontinuierlich aufsummiert werden, muss es später möglich sein, sie wieder zu subtrahieren.

Bei der Entwicklung des fullSubtractor() konnte man sich von VHDL Programmen, die frei verfügbar sind, inspirieren lassen. Es ist zuerst ein einfaches C++ Programm erstellt worden, um den Subtractor besser zu verstehen. Das C++ Programm hat auch geholfen, die homomorphe Variante umzusetzen, da die Implementierungen der beiden Varianten in C++ ähnlicher sind.
Die Umsetzung des fullSubtractor wurde aus Zeitgründen und wegen der niedrige Priorisierung nicht fertiggestellt.

5.4.2 Verifizierung

Die vorhandene C Library hat bereits GoogleTest Unit-Tests und Integrations-Tests. Die Tests sind als CUTE-Tests Stück um Stück umgeschrieben worden. Die automatische Umwandlung der Tests mit dem am IFS entwickelten Eclipse Plugin von GoogleTests zu CUTE-Tests ist nicht möglich, da das Plugin noch einige Limitationen hat. Der Hauptgrund ist, dass die vorhandenen GoogleTests nicht umgewandelt werden konnten, da keine Umwandlungen von Test-Fixtures (TEST_F) unterstützt werden.
Die Funktionalität der verschiedenen Applikationen sind mit dem vom Dozenten zur Verfügungestellten angepassten Source Code getestet worden. Die Schwierigkeit, das Programm zu prüfen, liegt in den verschlüsselten Daten. Der angepasste Source Code und die Applikationen können so schrittweise verglichen werden. Dabei spielt die Verschlüsselung der Daten keine Rolle, da beide gleich verschlüsselt werden müssen. Somit kann suggestive der Voting-Prozess überprüft werden. Dabei musste der Equals-Operator und der Output-Operator überschrieben werden.

5.5 Löschen der Secret-Keys im Memory

Die Private-Keys sind sensibel und sollten nicht von Dritten gelesen werden können. In dem Voting-System in der Core-Library sind die LweKey und die TLweKey Klasse sensibel.

5.5.1 Memory Management

In C++ Programmen werden die Daten, welche in verschiedenen Typen abgespeichert werden können, normalerweise, wenn sie klein sind, auf dem Stack abgelegt und wenn sie grösser sind auf dem Heap. In C++ besteht die Möglichkeit das feingranular zu definieren. Der std::vector wird zum Teil auf dem Stack und auf dem Heap angelegt. Einfache Datentypen wie ints werden auf dem Stack abgelegt. Im Normalfall, wenn das Memory am Ende des Scopes freigegeben wird, werden die Daten dahinter nicht gelöscht. Der Memorybereich wird nur freigegeben, sodass der Speicher wieder verwendet werden kann.
Das Nicht-Löschen der Daten im Memory ist aus Performancegründen so umgesetzt. Bei sensiblen Daten wie Private-Keys stellt das ein Problem dar. Das Memory sollte mit Nullen überschrieben werden oder eventuell randomisiert werden. Das Randomisieren der Daten wird meist nicht gemacht, da das Generieren von Zufallswerten mehr Performance braucht. Die Randomisierung ist jedoch besser, da zum Beispiel beim Reverse-Engineering nicht so einfach auf den überschriebenen Speicherbereich geschlossen werden kann.
Das sensible Memory darf auch nicht geswapped werden, da dadurch die Daten auf einem permanenten Speicher abgelegt werden und auf diesen Speicher ohne grosse Berechtigungen zugegriffen werden kann. Das Verhindern von Swapping is betriebssystemabhängig. In der Umsetzung im Source Code wird die Adaption auf das Betriebssystem durch Pre-Compiler Statements gelöst.

5.5.2 Compiler Flags

Bei Release-Builds des Source Codes können Optimisierungsflags gesetzt werden. Je nach Grad der Optimierung (von O0 bis O3) wird mehr optimiert. Zusätzlich kann noch die LTO aktiviert werden. Die Optimierung stellt auch eine Herausforderungen dar, da der Compiler nicht benötigte Speicherzuweisungen, wie zum Beispiel das Überschreiben von Private-Keys mit Nullen, wegoptimieren kann. Es gibt compilerspezifische #pragmas, welche es erlauben, die Optimierung für gewisse Bereiche auszuschalten. In dem Source Code der Core-Library werden #pragmas für den g++ und clang++ verwendet. Vergleichbare Anweisungen für den Visual Studio Compiler sind weggelassen worden, da es sich nicht um eine priorisierte Plattform handelt, was so in den NFR definiert worden ist, zu Begin der Semesterarbeit. Die LTO ist bei den Implementierungen nicht beachtet worden, da dies Optimierungen zu Link Time gemacht wird. Dies macht es schwierig die Optimierung nachzuvollziehen und ob dann der Speicher mit sensiblen Daten wirklich überschrieben wird. Zudem

wurden keine Informationen gefunden, wie man die Funktion deaktivieren kann. Deswegen ist empfohlen, dass das Programm nicht mit LTO kompiliert wird, was im Projekt in der CMake-Datei auch nicht definiert ist.

5.5.3 Implementierungen

Zu Beginn der Arbeit ist ein C++ Template SecureDestroy erstellt worden. Das Template hat für einfache Datentypen auf dem Stack gut funktioniert. Das grosse Problem dabei ist, dass die eigentlichen sensiblen Daten auf dem Heap nicht gelöscht werden. Die Klassen mit den Private-Keys beinhalten Membervariablen mit std::vector, was das eigentliche Problem darstellt, da der std::vector seine Daten auf dem Heap ablegt.
Nach der Erkenntnis des Problems, und auch zuvor, wurden Problematiken mit Fachleuten vom IFS besprochen[3][4]. Beim Heap-Problem vom SecureDestroy gab gaben uns die Fachleute vom IFS kurz darauf eine Antwort mit Beispiel-Code mit einem NullingWrapper Template, welcher lösen sollte. Das Template musste noch erweitert werden, um die Art wie der Speicher genullt oder randomisiert wird. Es musste auch noch die Aktivierung und Deaktivierung vom Swapping eingebaut werden, welche aus dem SecureDestroy entnommen werden konnte. Des Weiteren musste der NullingWrapper noch weiter angepasst werden, sodass er funktionsfähig ist. Es hat ein Type-Conversion-Constructor gefehlt und die Assignment-Operators mussten angepasst werden. Eine Dereferenzierung wurde auch noch eingebaut, sodass es einfacher ist den std::vector zu ersetzen.
Der NullingWrapper kann wie im folgenden Beispiel Code, welcher aus der LweKey Klasse, beziehungsweise aus der Header-Datei, herauskopiert worden ist, eingesetzt werden.

```
1 std::vector<core::utils::NullingWrapper<int32_t>> key;
```

Die Optimierungshinweise für den Compiler müssen von ihm nicht eingehalten werden und können deshalb ignoriert werden. Dadurch, muss man bei jedem Release-Build das Programm decompilen und analysieren, um sicher zu stellen, dass der Speicher wirklich gelöscht wird. Auch wenn der Compiler das Programm nicht optimiert hat, kann es immer noch sein, dass der CPU bei der Ausführung die Instruktionen optimiert und dadurch der Speicher nicht überschrieben wird.

5.5.4 Messung

Der NullingWrapper ist in einem kleinen Programm analysiert worden. Im Programm werden 100'000 Elemente an einem std::vector angefügt. Die Messung wird 100 Mal durchgeführt und dann wird der Mittelwert aller Messungen berechnet.
Der Test wird unter folgenden Bedingungen getestet:

- Messtestaufbau
  - Der Test wird als Release-Build mit dem Optimierungsflag -O3 kompiliert
  - Der Test ist mit C++17 Language Support kompiliert
  - Der Testcomputer hat einen Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz
  - Es wurden alle anderen Programme geschlossen, währenddessen die Messungen durchgeführt worden sind
- Limitationen
  - Kein Realtimesystem, somit haben andere Prozesse Einfluss auf die Messung

Es werden folgende Szenarien getestet:

- Vector ohne NullingWrapper
  - Es wird ein normaler std::vector mit int32_t Werten erstellt und dann durch den standard Destructor gelöscht.
- Vector mit einfachem NullingWrapper, Werte werden genullt.

	Ohne NullingWrapper	Vector mitNullingWrapper; Werte genullt	VectorNullingWrapper; Werte randomisiert	OptimierterNullingWrapper; Werte genullt
Ganze Funktion (μs)	399	428506	14656463	425000
Funktion im Vergleich zu ohne NullingWrapper $(-)$	1	1073.94	36732.98	1065.16
Destructor (μs)	21	43255	4333657	42227
Destructor im Vergleich zu ohne NullingWrapper $(-)$	1	2059.76	206364.61	2010.80

Tabelle 23: NullingWrapper Auswertung der verschiedenen Szenarien; 100'000 Elemente im std::vector; Mittelwert von je 100 Messungen

  - Es wird ein normaler std::vector mit int32_t Werten erstellt und die int32_t dem NullingWrapper umhüllt, was das Nullen der Werte bewirkt.

- Vector mit einfachem NullingWrapper, Werte werden randomisiert
  - Es wird ein normaler std::vector mit int32_t Werten erstellt und die int32_t dem NullingWrapper umhüllt, was das Randomisieren der Werte bewirkt, da zusätzlich noch das Flag gesetzt wird.

- Vector mit optimiertem NullingWrapper, Werte werden genullt.
  - Er unterscheidet sich zum normalen NullingWrapper, dadurch, dass grössere Werte als std::byte geschrieben werden. Es wird sizeof(int) auf einmal geschrieben, was auf den meisten Plattformen 4 Byte entspricht.

Die Testresultate sind in der folgenden Tabelle aufgeführt. Die Zeitwerte sind alle in Mikrosekunden (μs) angegeben.
Der Unterschied vom std::vector ohne NullingWrapper und mit NullingWrapper sind beachtlich. Die Zeiten vom normal NullingWrapper beträgt ungefähr 429ms für die ganze Funktion und 43ms für den ganzen Destructor, bei 100'000 Elementen im Vector, was ein erträglicher Wert ist.
Wenn Zufallswerte anstatt Nullen in den Speicher geschrieben werden, braucht das erheblich länger. Dies hat mit dem Generieren der Zufallswerte zu tun. Diese Variante wird deswegen nicht im E-Voting-System verwendet.

5.5.5 Optimierter NullingWrapper

Der optimierte NullingWrapper ist geringfügig schneller. Der Hauptunterschied der beiden NullingWrapper Varianten ist, dass der optimierte pro Nullung die Grösse von sizeof(int) setzt und der normale nur die Grösse von std::byte. Dies ist unten im decomplierten Code ersichtlich. Der normale muss 4 Mal Nullen (Erster Codeabschnitt, Zeile 9, while-Loop) und der optimierte muss nur 1 Mal Nullen (Zweiter Codeabschnitt, Zeile 11, while-Loop). Die maximale Schreibgrösse pro Instruktion ist nicht bestimmbar mit C++, die Grösse vom int ist die bestmögliche Approximation. Dies hat mit dem Abstraktionslevel von C++ zutun. Es ist nicht ideal, dass die meisten C++ Compiler int als 32-bit Wert übersetzen und dies auch auf 64-bit Systemen. Somit ist bei den meisten Systemen das Maximum das zeitgleich genullt werden kann 32-bit. In diesem Test ist nur ein kleiner Wert genullt worden. Bei grösseren Werten müsste der optimierte NullingWrapper erheblich schneller sein.
Decomplierter Code vom normalen NullingWrapper:

```
1  /* core::utils::NullingWrapper<int>::scribble() */
2  void _ZN4core5utils14NullingWrapperIiE8scribbleEv(struct s0* rdi, struct s0* rsi, struct s0* rdx) {
3      struct s0* v4;
4      int32_t v5;
5      signed char* rax6;
6
7      v4 = rdi;
8      v5 = 0;
9      while (v5 <= 3) {
10         rax6 = _ZNSt5arrayISt4byteLm4EEixEm(v4, static_cast<int64_t>(v5));
11         *rax6 = 0;
12         ++v5;
13     }
14     return;
15 }
```

Decomplierter Code vom optimierten NullingWrapper:

```
/* core::utils::NullingWrapper<int>::scribble() */
void _ZN4core5utils14NullingWrapperIiE8scribbleEv(struct s0* rdi, struct s0* rsi, struct s0* rdx) {
    struct s0* rax4;
    struct s0* v5;
    int32_t v6;
    int32_t v7;

    rax4 = _ZNSt5arrayISt4byteLm4EE4dataEv(rdi, rsi);
    v5 = rax4;
    v6 = 0;
    while (v6 <= 0) {
        *reinterpret_cast<struct s0**>(&v5->f0) = reinterpret_cast<struct s0*>(0);
        v5 = reinterpret_cast<struct s0*>(&v5->f4);
        ++v6;
    }
    if (!1) {
        v7 = 0;
        while (v7 < 0) {
            *reinterpret_cast<struct s0**>(&v5->f0) = reinterpret_cast<struct s0*>(0);
            ++v7;
        }
    }
    return;
}
```

Der NullingWrapper sollte die Werte überschreiben. Mit Caching kann es jedoch sein, dass die Werte immer noch vorhanden sind. Dies zu lösen ist plattformabhängig und wird nicht umgesetzt. In den Anforderungsspezifikationen bei den NFR ist eine hohe Plattformunabhängigkeit definiert worden, was sich mit einer hardwareabhängigen Lösung ohne grossen Aufwand nicht verträgt.

5.5.6 Weitere Lösungsmöglichkeiten

Eine weitere Optimierungsmöglichkeit wäre, eine ADT zu programmieren, welche man NullingVector nennen könnte. Der NullingVector hat folgende Vor- und Nachteile:

- Vorteile
  - Beim Erstellen und Löschen der Datenstruktur ergeben sich weniger Instruktionen, da in int-Grössen Speicher überschrieben und über den ganzen Speicher iteriert werden kann. Beim NullingWrapper wird für jedes Element der Destructor aufgerufen.
- Nachteile
  - Der NullingVector kann nur für Vectoren gebraucht werden, wobei der NullingVector universal für verschiedene ADT verwendet werden kann.

Der NullingVector ist aus zeitlichen Gründen nicht implementiert worden. Die oben erwähnten Vor- und Nachteile müssten genauer evaluiert werden, wenn ein NullingVector umgesetzt worden ist.
Die Korrektheit des NullingWrapper konnte über Decompling verfiziert werden, jedoch nicht formell. Eine Programmiersprache wie Ada hätte die Möglichkeit dazu und wäre somit besser geeignet.

6 Schlussfolgerung

6.1 Zentrale Herausforderungen

6.1.1 Mathematik

Ein wichtiger Bestandteil der Arbeit war die Benutzung und Portierung der bestehenden TFHE Library auf C++17. Die recht komplexe Funktionalität zu verstehen stellte uns spezeill bei der Fehlersuche vor grosse Herausforderungen.

6.2 Ist-Soll Vergleich

6.2.1 Abuse-Cases

Auslastung des Systems
Wie in dem Kapitel 6.2.3 beschrieben wird, ist es möglich mehrere Bulletin Boards zu erzeugen. Das Auszählen der Stimmen kann unabhänig von einander durchgeführt werden. Bei Überlastung eines Bulletin Boards können die anderen immer noch die Ergebnisse produzieren und an den Coordinator weitergeben. Im entwickelten E-Voting-System gibt es pro Wahl nur ein Coordinator, was immer noch zu einem Single Point of Failure führt. In diesem Prototyp müsste der Coordinator mit einer Firewall geschützt werden, der DDoS Angriffe abwenden kann.

Datenkorruption
Im Prototyp des E-Voting-System befindet sich die einzige Inputstelle im Coordinator. Da es sich um einen Prototypen handelt, wurde der Bereich der Zahlenwerte nicht begrenzt.

Parallele Systeme zum Abstimmen
Der Prototyp beinhaltet keine Netzwerkfähigkeit, dadruch befindet sich das Generieren der Ballots auf dem selben Computer wie das Zusammenzählen. Es ist nicht möglich, von aussen falsche Stimmzettel hinzuzufügen.

Erstellen einer neuen Wahl
Das Abuse-Case ist organisatorischer Natur, deshalb ist es beim Prototypen nicht umgesetzt worden.

Beeinflussung der Wähler
Es wurde begonnen einen fullSubtractor zu implementieren, der Stimmen bei Bedarf wieder abzählen kann. Die Umsetzung hat sich als nicht trivial gestaltet und wurde dadurch nicht fertig umgesetzt. Eine genauere Beschreibung ist im Kapitel 5.4.1 zu finden.

Entschlüsselung von Fake Daten
Die Verifizierung der Daten kann gewährleistet werden, in dem mehrere Trustees 6.2.3 und mehrere Bulletin Boards 6.2.3 bestehen.

Nachvollziehbarkeit der Wahlresultate
Die Nachvollziehbarkeit der Wahlresultate können mithilfe der Daten, die am Ende des Abstimmungsprozesses veröffentlicht werden, von Dritten verifiziert werden.

Trends der Abstimmung erkennen
Das Berechnen der Wahlergebnisse und diese im Klartext zu sehen, kann zur Beeinflussung der Wahl verwendet werden, aber durch den Einsatz von mehreren Trustees 6.2.3 kann die Entschlüsselung verhindert werden, wenn nicht alle Entities zusammenarbeiten, die einen Trustee betreiben.

6.2.2 Nicht Funktionale Anforderungen

Randbedingungen
Die Randbedingungen sind wie in den nicht funktionalen Anforderungen definiert umgesetzt worden. Zudem

ist der Source Code vom betreuenden Dozenten dazugezogen geworden. Der fullAdder() der vom Team entwickelt worden ist, wurde zusätzlich hinzugefügt.

Usability
Wie schon in den Anforderungsspezifikationen erwähnt worden ist, die Usability niedrig priorisiert worden. Das E-Voting-System kann in der Console ausgeführt werden. Darauf folgend können die Parameter eingegeben werden und dann kann der automatisierte Voting-Prozess gestartet werden. Die nähere Beschreibung wie das E-Voting-System bedient werden kann ist im Kapitel G zu finden.

Privacy
Die Privacy, die in den nicht funktionalen Anforderungen definiert worden sind können eingehalten werden. Detailliertere Informationen können dem Kapitel 5.5 entnommen werden.

Security
Der Prototyp besitzt keine Netzwerkfähigkeit, deshalb wurden keine besonderen Massnahmen zur Abwedung von Angriffen getroffen.

Plattform
Der Plattformsupport unter Linux ist gewährleistet, da die Entwicklung mehrheitlich auf dieser Plattform stattgefunden hat. Die Applikation kann auch unter Windows mit WSL (getestet mit Ubuntu) ausgeführt werden. Der OSX Support konnte nicht überprüft werden, da kein Computer mit diesem Betriebssystem zur Verfügung stand. Bei der Implementierungen ist auf die Plattformunabhängigkeit geachtet worden, deshalb sollte die Applikation ohne Probleme auf dem System lauffähig sein. Die Applikation muss neu kompiliert werden auf einem OSX-Rechner, sodass sie lauffähig ist.

Performance
Tests haben ergeben, dass das E-Voting-System geringfügig länger braucht, um die Stimmen zusammen zu zählen als, die originale Version. Detailinformationen können dem Kapitel 5.3 entnommen werden.
Der 'Leveled mode LHE' konnte nicht implementiert werden. Die Funktionalität ist nicht in der originalen Library enthalten. Die Kapazität, diese Funktionalität zu integrieren, war nicht vorhanden.

6.2.3 Aufgabenstellung

N Kandidaten Abstimmung (ink. Stimmenthaltung)
Unser Prototyp kann Abstimmungen mit 2 oder mehr Wahlmöglichkeiten durchführen, eine Ja/Nein/Enthaltung Abstimmung benötigt 3 Wahlmöglichkeiten. Intern verwendet der Prototyp für diese Wahlmöglichkeiten die nächsthöhere Potenz der Zahl 2 (z.B. 4) alle nicht benötigten Möglichkeiten werden als Stimmenthaltung gewertet.

LweKeys auf mehreren Trustees
Grundsätzlich ist es möglich die LweKeys auf mehrere Trustees zu verteilen. Im laufe der Arbeit ist uns jedoch ein Fehler in der Implementation des Papers aufgefallen. Dieser Fehler wurde nun im Paper angepasst es war uns aber zeitlich nicht mehr möglich diese Änderung zu berücksichtigen.

Mehrere Bulletin Boards
Grundsätzlich kann es in unserer beliebig viele Bulletin Boards geben es wird aber zunehmend schwierig es zu testen, da der Ressourcenverbrauch stark ansteigt. Da sich die Bulletin Boards in der Produktion auf mehrere Rechner aufteilen sollte das aber kein problem sein.

6.2.4 Verbesserungswürdige Punkte im Rückblick

Es gibt verbesserungswürdige Punkte im funktionalen und architektonischen Bereich. Der fullSubtractor() ist ein funktionaler Punkt und die weiteren Punkte sind architektonisch bedingt.

Full Subtractor
Die fullSubtractor() Funktionalität, die im Kapitel 5.4.1 beschrieben ist, könnte in einer Folgearbeit umgesetzt werden. Der Restaufwand sollte nicht zu gross sein, da mit der Implementierungen schon begonnen wurde.

Multithreading
Im jetzigen Zustand hat die Applikation nur begrenzte Multithreading Fähigkeiten. Dies könnte in einer Folgearbeit weiter verbessert werden, um eine Performance Steigerung zu erlangen.

Smart-Pointer
Die Daten könnten per Smart-Pointer übergeben werden, sodass keine unnötigen Kopien erstellt werden müssen. Der Vorteil gegenüber Referenzen wären, das man sich nicht damit befassen muss, ob die referenzierten Daten noch existieren.

GPU Beschleuningung
Die Matrizenrechnungen könnten auf den GPU ausgelagert werden, um so potentiell das Resulat schneller zu berechnen. So wäre es potentiell möglich, den fullAdder() auf die GPU auszulagern. Die Frage ist jedoch, ob der ganze KeySwitchingKey im Memory auf der Grafikkarte Platz hat. Das Kopieren der Daten auf das Grafikkartenmemory kann dabei zum Bottleneck führen, deswegen müsste dies ausgetauscht werden.

6.3 Weiteres Vorgehen

Netzwerkfähigkeit
In der jetztigen Form ist das E-Voting-System eine einzelne Applikation. Die Schnittstellen zwischen dem Coordinator, Trustee, Bulletin Board und dem Voter sind über Pure Virtual Functions (Interfaces in Java) definiert. Das E-Voting-System könnte mithilfe von Boost.Asio oder einer RPC Library für C++ netzwerktauglich gemacht werden.
Wenn die Daten über das Netwerk übertragen werden sollten, sollten vor allem aus Privacy Gründen die Daten mit TLS 1.3 verschlüsselt werden. Es könnten auch VPN Verbindungen zwischen den verschiedenen Endpunkten aufgebaut werden, was die gleichen oben genannten Problematiken beseitigen könnte. Dabei wird vermutlich das Zusammenspiel schwieriger, jedoch ist die Verbindung der Endpunkte einfacher verifizierbar.

Signierung der Entities
In den Anforderungsspezifikationen ist definiert worden, dass das Verifizieren der einzelnen Teilsystemen des E-Voting-Systems in dieser Semesterarbeit nicht beachtet wird. Eine Folgearbeit müsste Signaturen einführen um die verschiedenen Endpunkte eindeutig unterscheiden zu können.

Separierung LWE und TLWE Keys
In der geschriebenen Applikation ist es möglich aus dem Cloud-Key, die Private Keys zurück zu gewinnen, was die Verschlüsselung unsicher macht. Dies ist in der neuen veröffentlichte Arbeit [1] gelöst worden. Mithilfe strikter Separierung der LWE und TWLE Keys auf die einzelnen Trustees mittels der Multi-Key Erweiterung ist möglich [5] das E-Voting-System sicher zu machen.

7 Projekt Management

7.1 Arbeitszeiten

Währen der Projektarbeit haben wir unsere Arbeitszeiten erfasst und kategorisiert. Aus den erfassten Daten konnten wir eine Statistik der Arbeitsstunden erstellen. Dabei haben wir die Arbeitszeiten in folgende Kategorien eingeteilt:

Feature
Features beinhaltet die Software Entwicklung sowie die Dokumentation.

Planung
Erarbeiten der Funktionalen und nicht Funktionalen Anforderungen.

Team Sitzung
Planung der Sprints, Sitzungen mit dem Betreuer und Reviews.

Support
Unterstützende arbeiten wie Einrichtung der Entwicklungsumgebung.

Bug
Beheben von Fehlern in abgeschlossenen Arbeitspaketen.

Chore
Initiale Einrichtung des Buildservers und der Programmierumgebung sowie Redmine und Gitlab.

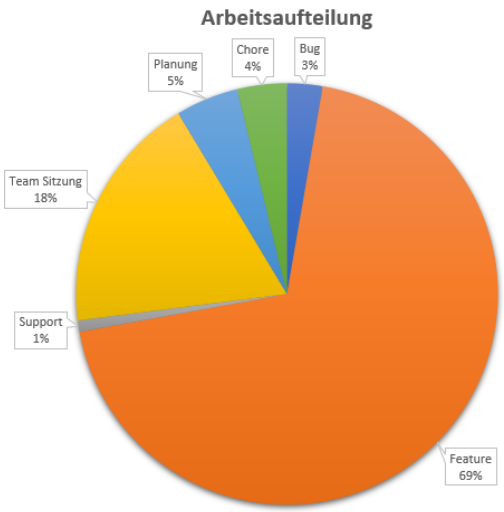
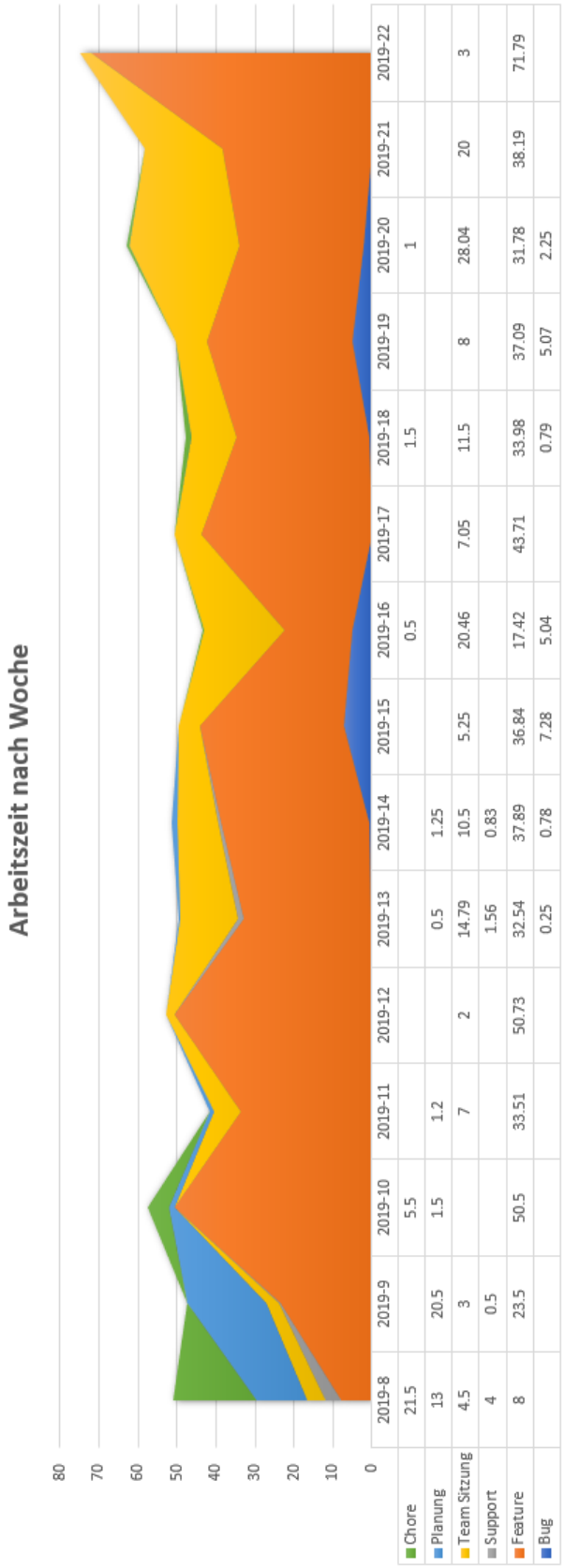


Abbildung 5: Arbeitsaufteilung



	2019-8	2019-9	2019-10	2019-11	2019-12	2019-13	2019-14	2019-15	2019-16	2019-17	2019-18	2019-19	2019-20	2019-21	2019-22
Chore	21.5		5.5						0.5		1.5		1		
Planung	13	20.5	1.5	1.2		0.5	1.25								
Team Sitzung	4.5	3		7	2	14.79	10.5	5.25	20.46	7.05	11.5	8	28.04	20	3
Support	4	0.5				1.56	0.83								
Feature	8	23.5	50.5	33.51	50.73	32.54	37.89	36.84	17.42	43.71	33.98	37.09	31.78	38.19	71.79
Bug						0.25	0.78	7.28	5.04		0.79	5.07	2.25		

Abbildung 6: Arbeitszeiten

Zeitschätzung
Zu den Arbeitspaketen haben wir vor Beginn jeweils eine Zeitschätzung erfasst. Diese haben wir nach Abschluss des Pakets mit der tatsächlichen Zeit verglichen. In den meisten Fällen waren die Schätzungen im Rahmen von 15% Abweichung. Zum Teil waren die Schätzungen zu hoch angesetzt und das Arbeitspaket wurden früher fertig. Etwas weniger Schätzungen waren zu niedrig angesetzt, wobei dort die Streuung grösser ist. Sporadisch wurden auch Pakete ohne Schätzung bearbeitet, das ist als -100% Abweichung in der Statistik.

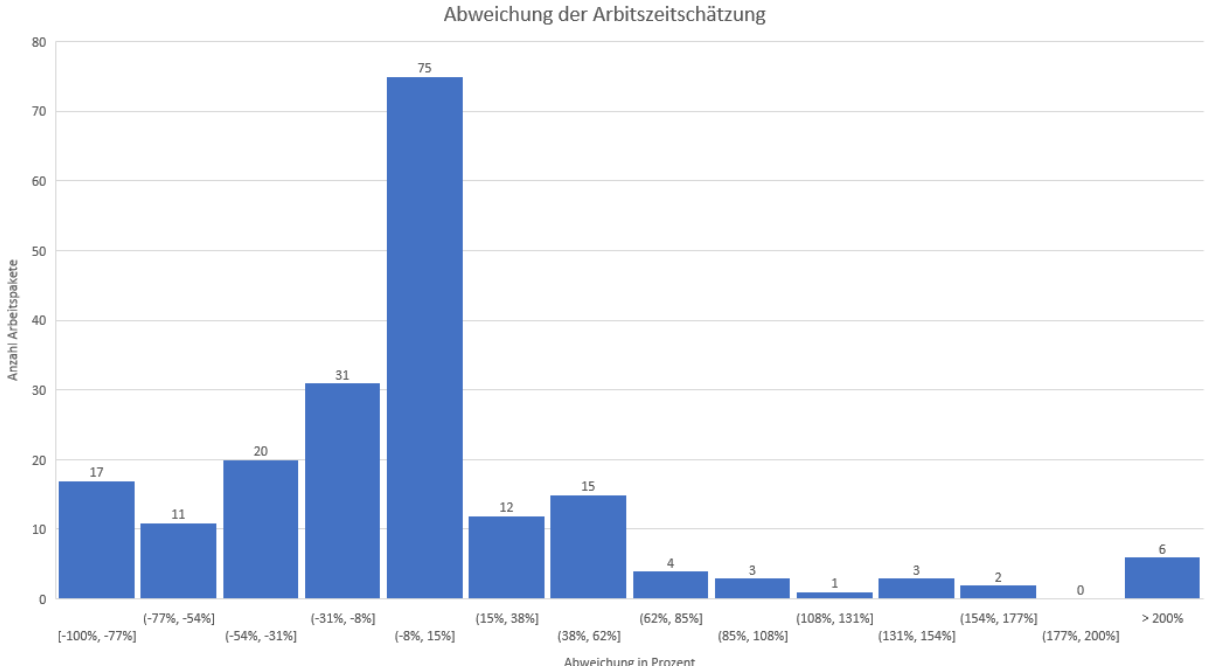
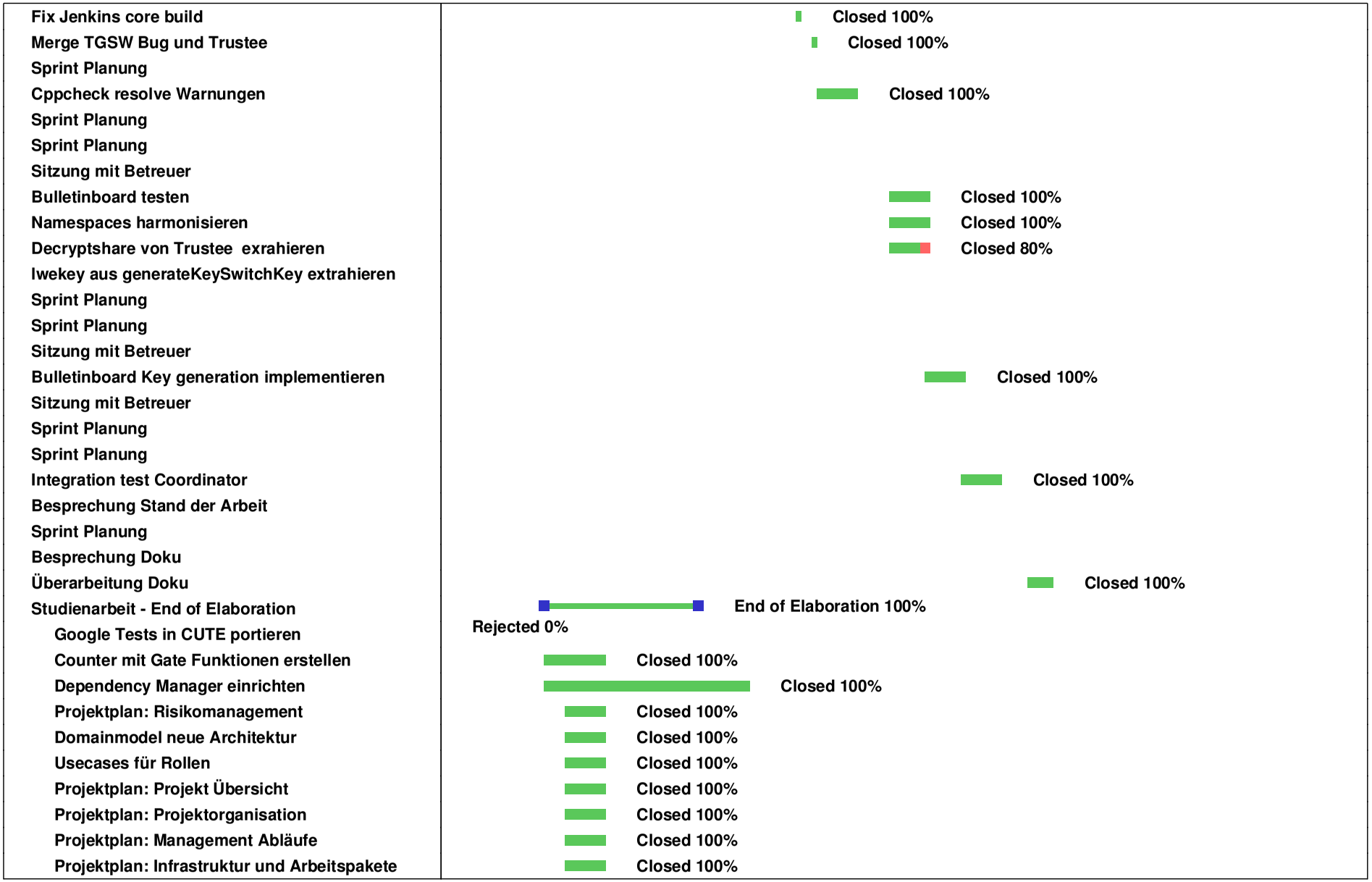
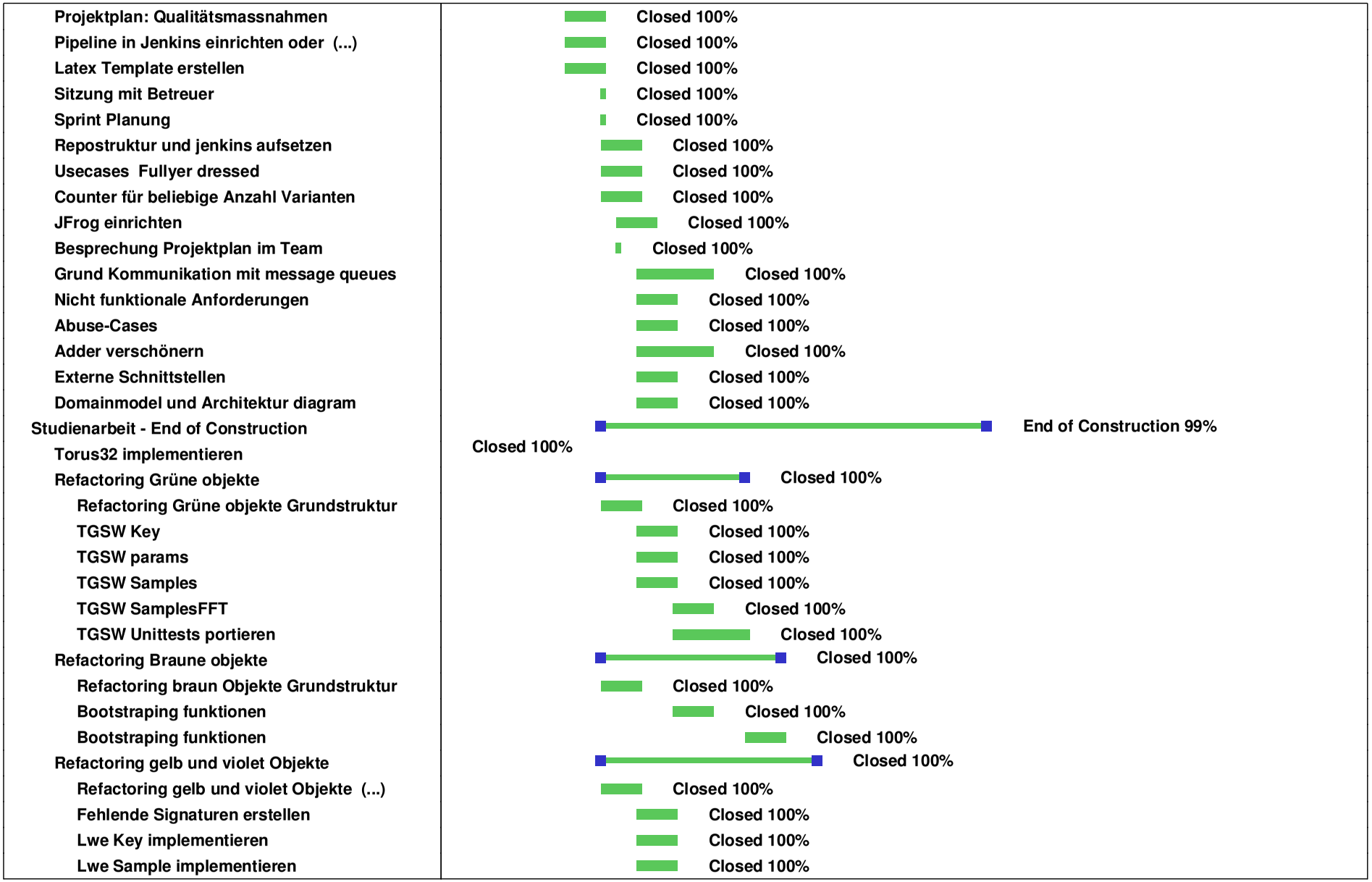
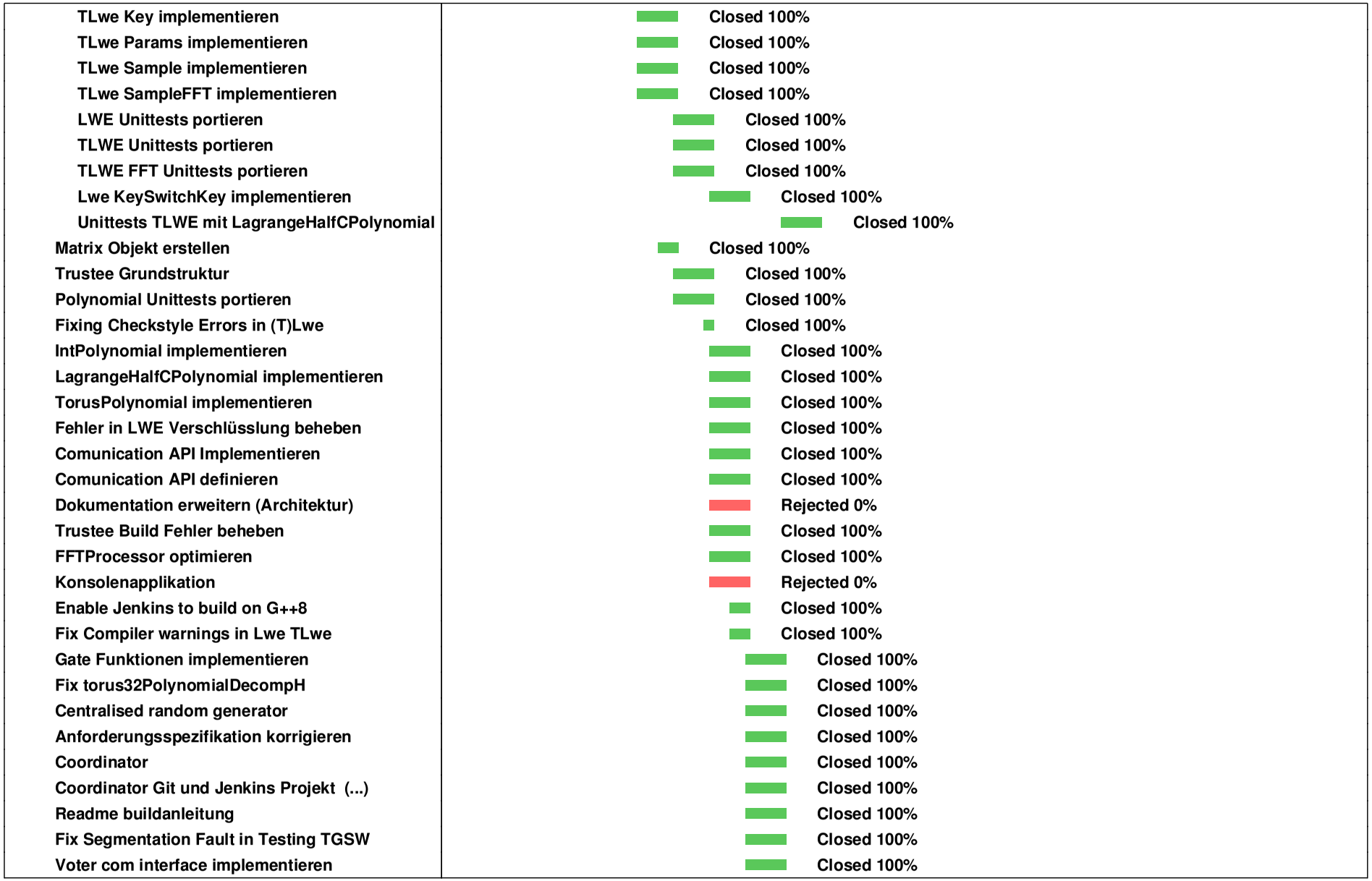


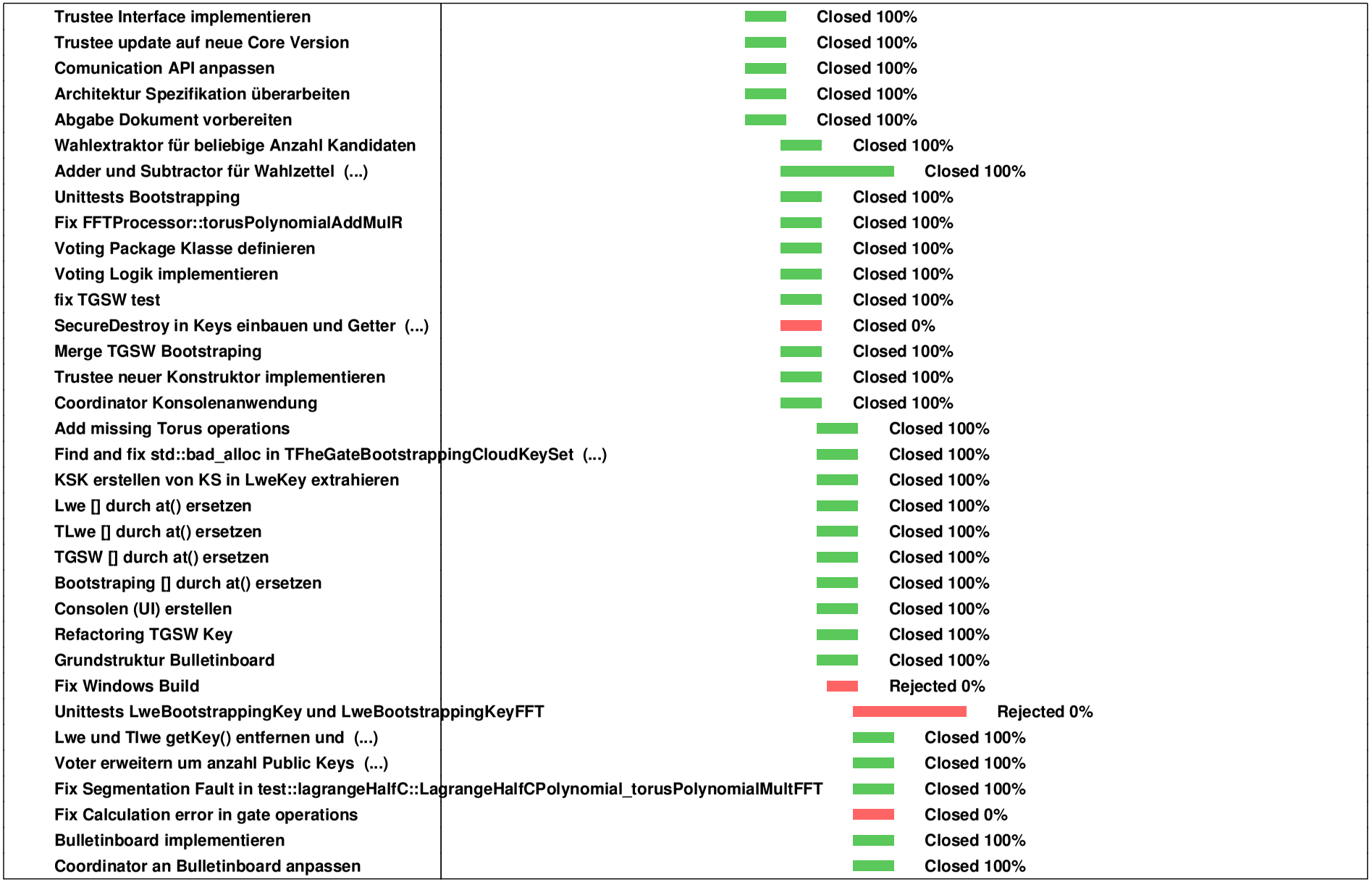
Abbildung 7: Abweichungen der Zeitschätzungen

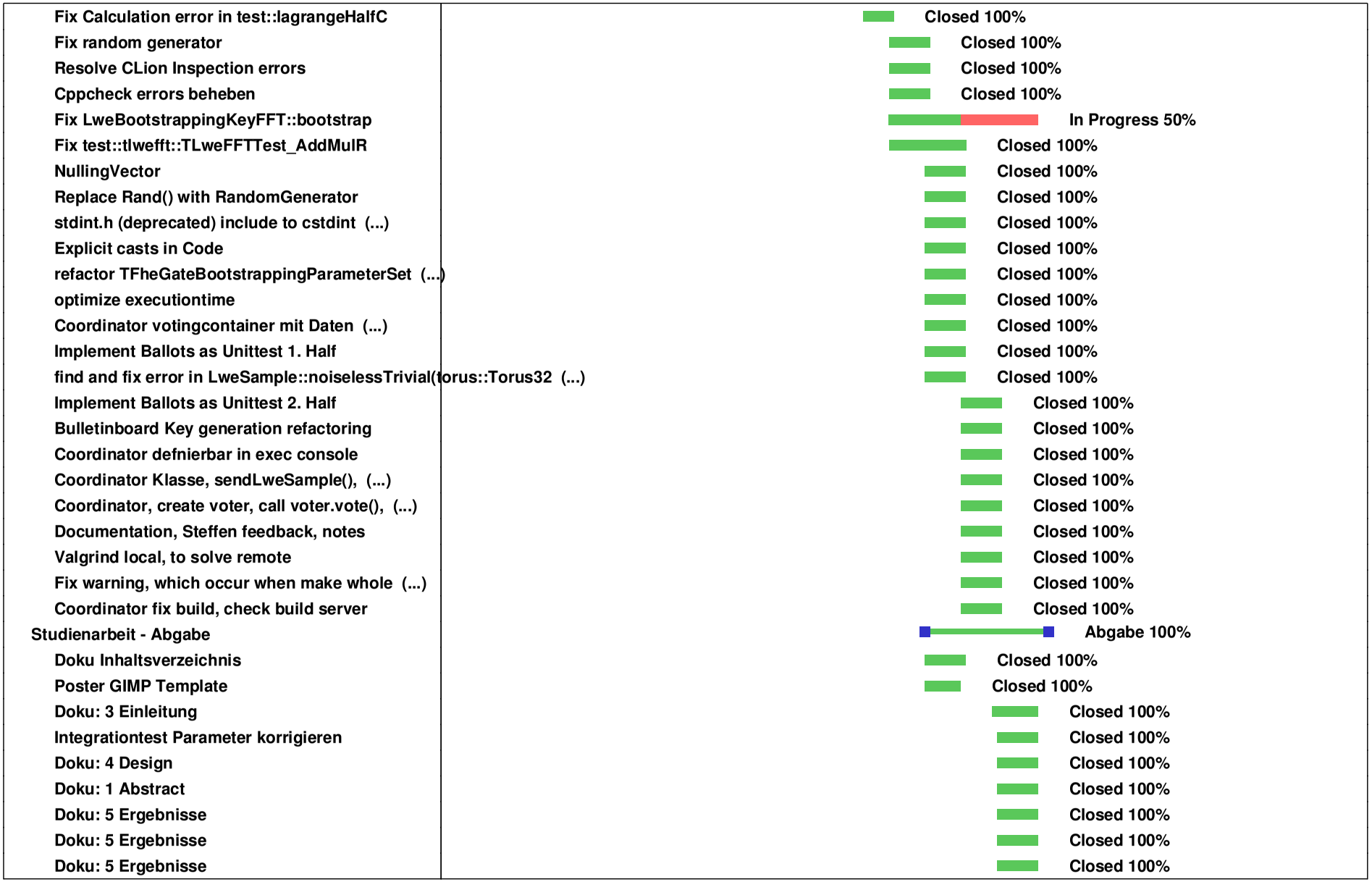
Studienarbeit


Fix Jenkins core build
Closed 100%
Merge TGSW Bug und Trustee
Closed 100%
Sprint Planung
Cppcheck resolve Warnungen
Closed 100%
Sprint Planung
Sprint Planung
Sitzung mit Betreuer
Bulletinboard testen
Closed 100%
Namespaces harmonisieren
Closed 100%
Decryptshare von Trustee exrahieren
Closed 80%
Iwekey aus generateKeySwitchKey extrahieren
Sprint Planung
Sprint Planung
Sitzung mit Betreuer
Bulletinboard Key generation implementieren
Closed 100%
Sitzung mit Betreuer
Sprint Planung
Sprint Planung
Integration test Coordinator
Closed 100%
Besprechung Stand der Arbeit
Sprint Planung
Besprechung Doku
Überarbeitung Doku
Closed 100%
Studienarbeit - End of Elaboration
End of Elaboration 100%
Google Tests in CUTE portieren
Rejected 0%
Counter mit Gate Funktionen erstellen
Closed 100%
Dependency Manager einrichten
Closed 100%
Projektplan: Risikomanagement
Closed 100%
Domainmodel neue Architektur
Closed 100%
Usecases für Rollen
Closed 100%
Projektplan: Projekt Übersicht
Closed 100%
Projektplan: Projektorganisation
Closed 100%
Projektplan: Management Abläufe
Closed 100%
Projektplan: Infrastruktur und Arbeitspakete
Closed 100%


Projektplan: Qualitätsmassnahmen
Closed 100%
Pipeline in Jenkins einrichten oder (...)
Closed 100%
Latex Template erstellen
Closed 100%
Sitzung mit Betreuer
Closed 100%
Sprint Planung
Closed 100%
Repostruktur und jenkins aufsetzen
Closed 100%
Usecases Fullyer dressed
Closed 100%
Counter für beliebige Anzahl Varianten
Closed 100%
JFrog einrichten
Closed 100%
Besprechung Projektplan im Team
Closed 100%
Grund Kommunikation mit message queues
Closed 100%
Nicht funktionale Anforderungen
Closed 100%
Abuse-Cases
Closed 100%
Adder verschönern
Closed 100%
Externe Schnittstellen
Closed 100%
Domainmodel und Architektur diagram
Closed 100%
Studienarbeit - End of Construction
End of Construction 99%
Torus32 implementieren
Closed 100%
Refactoring Grüne objekte
Closed 100%
Refactoring Grüne objekte Grundstruktur
Closed 100%
TGSW Key
Closed 100%
TGSW params
Closed 100%
TGSW Samples
Closed 100%
TGSW SamplesFFT
Closed 100%
TGSW Unittests portieren
Closed 100%
Refactoring Braune objekte
Closed 100%
Refactoring braun Objekte Grundstruktur
Closed 100%
Bootstraping funktionen
Closed 100%
Bootstraping funktionen
Closed 100%
Refactoring gelb und violet Objekte
Closed 100%
Refactoring gelb und violet Objekte (...)
Closed 100%
Fehlende Signaturen erstellen
Closed 100%
Lwe Key implementieren
Closed 100%
Lwe Sample implementieren
Closed 100%


TLwe Key implementieren
Closed 100%
TLwe Params implementieren
Closed 100%
TLwe Sample implementieren
Closed 100%
TLwe SampleFFT implementieren
Closed 100%
LWE Unittests portieren
Closed 100%
TLWE Unittests portieren
Closed 100%
TLWE FFT Unittests portieren
Closed 100%
Lwe KeySwitchKey implementieren
Closed 100%
Unittests TLWE mit LagrangeHalfCPolynomial
Closed 100%
Matrix Objekt erstellen
Closed 100%
Trustee Grundstruktur
Closed 100%
Polynomial Unittests portieren
Closed 100%
Fixing Checkstyle Errors in (T)Lwe
Closed 100%
IntPolynomial implementieren
Closed 100%
LagrangeHalfCPolynomial implementieren
Closed 100%
TorusPolynomial implementieren
Closed 100%
Fehler in LWE Verschlüsslung beheben
Closed 100%
Comunication API Implementieren
Closed 100%
Comunication API definieren
Closed 100%
Dokumentation erweitern (Architektur)
Rejected 0%
Trustee Build Fehler beheben
Closed 100%
FFTProcessor optimieren
Closed 100%
Konsolenapplikation
Rejected 0%
Enable Jenkins to build on G++8
Closed 100%
Fix Compiler warnings in Lwe TLwe
Closed 100%
Gate Funktionen implementieren
Closed 100%
Fix torus32PolynomialDecompH
Closed 100%
Centralised random generator
Closed 100%
Anforderungsspezifikation korrigieren
Closed 100%
Coordinator
Closed 100%
Coordinator Git und Jenkins Projekt (...)
Closed 100%
Readme buildanleitung
Closed 100%
Fix Segmentation Fault in Testing TGSW
Closed 100%
Voter com interface implementieren
Closed 100%


Trustee Interface implementieren
Closed 100%
Trustee update auf neue Core Version
Closed 100%
Comunication API anpassen
Closed 100%
Architektur Spezifikation überarbeiten
Closed 100%
Abgabe Dokument vorbereiten
Closed 100%
Wahlextraktor für beliebige Anzahl Kandidaten
Closed 100%
Adder und Subtractor für Wahlzettel (...)
Closed 100%
Unittests Bootstrapping
Closed 100%
Fix FFTProcessor::torusPolynomialAddMulR
Closed 100%
Voting Package Klasse definieren
Closed 100%
Voting Logik implementieren
Closed 100%
fix TGSW test
Closed 100%
SecureDestroy in Keys einbauen und Getter (...)
Closed 0%
Merge TGSW Bootstraping
Closed 100%
Trustee neuer Konstruktor implementieren
Closed 100%
Coordinator Konsolenanwendung
Closed 100%
Add missing Torus operations
Closed 100%
Find and fix std::bad_alloc in TFheGateBootstrappingCloudKeySet (...)
Closed 100%
KSK erstellen von KS in LweKey extrahieren
Closed 100%
Lwe [] durch at() ersetzen
Closed 100%
TLwe [] durch at() ersetzen
Closed 100%
TGSW [] durch at() ersetzen
Closed 100%
Bootstraping [] durch at() ersetzen
Closed 100%
Consolen (UI) erstellen
Closed 100%
Refactoring TGSW Key
Closed 100%
Grundstruktur Bulletinboard
Closed 100%
Fix Windows Build
Rejected 0%
Unittests LweBootstrappingKey und LweBootstrappingKeyFFT
Rejected 0%
Lwe und Tlwe getKey() entfernen und (...)
Closed 100%
Voter erweitern um anzahl Public Keys (...)
Closed 100%
Fix Segmentation Fault in test::lagrangeHalfC::LagrangeHalfCPolynomial_torusPolynomialMultFFT
Closed 100%
Fix Calculation error in gate operations
Closed 0%
Bulletinboard implementieren
Closed 100%
Coordinator an Bulletinboard anpassen
Closed 100%


Fix Calculation error in test::lagrangeHalfC
Closed 100%
Fix random generator
Closed 100%
Resolve CLion Inspection errors
Closed 100%
Cppcheck errors beheben
Closed 100%
Fix LweBootstrappingKeyFFT::bootstrap
In Progress 50%
Fix test::tlwefft::TLweFFTTest_AddMulR
Closed 100%
NullingVector
Closed 100%
Replace Rand() with RandomGenerator
Closed 100%
stdint.h (deprecated) include to cstdint (...)
Closed 100%
Explicit casts in Code
Closed 100%
refactor TFheGateBootstrappingParameterSet (...)
Closed 100%
optimize executiontime
Closed 100%
Coordinator votingcontainer mit Daten (...)
Closed 100%
Implement Ballots as Unittest 1. Half
Closed 100%
find and fix error in LweSample::noiselessTrivial(torus::Torus32 (...)
Closed 100%
Implement Ballots as Unittest 2. Half
Closed 100%
Bulletinboard Key generation refactoring
Closed 100%
Coordinator defnierbar in exec console
Closed 100%
Coordinator Klasse, sendLweSample(), (...)
Closed 100%
Coordinator, create voter, call voter.vote(), (...)
Closed 100%
Documentation, Steffen feedback, notes
Closed 100%
Valgrind local, to solve remote
Closed 100%
Fix warning, which occur when make whole (...)
Closed 100%
Coordinator fix build, check build server
Closed 100%
Studienarbeit - Abgabe
Abgabe 100%
Doku Inhaltsverzeichnis
Closed 100%
Poster GIMP Template
Closed 100%
Doku: 3 Einleitung
Closed 100%
Integrationtest Parameter korrigieren
Closed 100%
Doku: 4 Design
Closed 100%
Doku: 1 Abstract
Closed 100%
Doku: 5 Ergebnisse
Closed 100%
Doku: 5 Ergebnisse
Closed 100%
Doku: 5 Ergebnisse
Closed 100%

Performance Tests		**Closed 100%**
Doku: 6.1 Zentrale Herausforderungen		**Closed 100%**
Doku: 6.2 Ist-Soll Vergleich		**Closed 100%**
Doku: Appendix B, Reflexionen Teammitglieder		**Closed 100%**

8 Abkürzungsverzeichnis

ADT Abstract Data Type. 15

DDoS Distributed Denial-of-Service. 16

LTO Link Time Optimization. 12, 13

LWE Learning with errors. 1–4, 6–8, 32

NFR Non Functional Requirements. 12, 15

TGSW Encryption by Craig **Gentry** and Amit **Sahai** and Brent **Waters**(GSW) over the **Torus**. 7

TLWE Learning with errors over the Torus. 7

VHDL VHSIC Hardware Description Language. 12

Quellenverzeichnis

[1] Hao Chen, Ilaria Chillotti und Yongsoo Song. *Multi-Key Homomophic Encryption from TFHE*. Cryptology ePrint Archive, Report 2019/116. https://eprint.iacr.org/2019/116. 2019.

[2] Ilaria Chillotti. *Towards efficient and secure Fully Homomorphic Encryption and cloud computing*. L'Université Paris-Sacla on Github. https://ilachill.github.io/papers/these_Ilaria_Chillotti_wo_acknowl.pdf. 2018.

[3] Thomas Corbat. *Memory Nulling with C++*. On different occations Thomas Corbat provided crucial information for the the SecureDestroy and NullingWrapper C++ Template for memory nulling. 2019. URL: https://www.hsr.ch/de/suche/personen/?q=Thomas+Corbat+.

[4] Felix Morgner. *Memory Nulling with C++*. Felix Morgner helped with the provided the sample code in coperation with Thomas Corbat. 2019. URL: https://www.hsr.ch/de/suche/personen/?q=Felix+Morgner+.

[5] Andreas Steffen. *Advisor Term Paper*. Andreas Steffen is the university Advisor of the term paper. 2019. URL: https://www.hsr.ch/de/suche/personen/?q=Steffen+Andreas.

Illustrationsverzeichnis

Tabellenverzeichnis

A Reflexion

A.0.1 Lukas Lätsch

Das Thema Post-Quantum-E-Voting hat mich sehr interessiert, auch wenn E-Voting ich eher kritisch gegenüber stehe. Die Möglichkeit, eine neuartige Verschlüsselung im Detail kennenzulernen und auch anzuwenden, war für mich eine gute Erfahrung. Das Einarbeiten in die Materie war für mich zu Beginn eher schwierig, insbesondere das Lesen des Papers. Durch das Lesen und Überarbeiten des Codes bin ich dann aber schliesslich gut an das Thema herangekommen. An das Programmieren in C++ habe ich mich relativ schnell gewöhnt. Dennoch hat sich das Überarbeiten des bestehenden Codes als zeitaufwändiger erwiesen, als ich zunächst erwartet hatte. Dafür gab es vorwiegend zwei Gründe. Zum einen ist die alt Bibliothek meiner Meinung nach eine unübersichtliche Mischung aus C und C++. Beispielsweise sind einige Klassen ausgestattet mit Konstruktoren und Destruktoren, während andere mittels Funktionen wie Structs initialisiert werden. Zudem waren die meisten Funktionen nicht an ein Objekt angehängt, obwohl sie es konzeptionell sein müssten. Zum anderen war es die Verwendung von C-Pointern anstelle von Smart-Pointern oder dem std::vecotr, die es teilweise sehr schwer machten, zu erkennen, ob es sich bei einer Variablen um ein Array oder einen Pointer handelt. Oft war es nötig, sich durch den Code zu kämpfen, um die Initialisierung der Variablen zu finden, bevor diese Frage beantwortet werden konnte. Die Planung und Durchführung des Projektes hat meiner Meinung nach gut funktioniert. Lediglich die Einteilung der Arbeitstakte war teilweise etwas schwierig, da es starke Abhängigkeiten zwischen verschiedenen Ebenen der Library gibt. Beispielsweise ist TGSW stark abhängig von LWE und TLWE. So gab es teilweise Verzögerungen, weil Abhängigkeiten noch pendent waren. Der vermutlich schwierigste Teil der Arbeit war aber die Fehlersuche mit verschlüsselten Daten. Beispielsweis entdeckten wir einen Fehler im keyswitch der sich als sehr schwer zu finden herausgestellt hatte. Ein Grund dafür ist, dass die verschlüsselten Daten nur wenig hilfreich sind um herauszufinden, wo sich der Fehler einschleicht, da sich falsche Zufallszahlen nicht wesentlich von den korrekten unterscheiden.

A.0.2 Rolf Furrer

Das Thema der Studienarbeit ist recht komplex, deswegen hat es einige Zeit gebraucht, bis man die Materie verstanden hat. Zugleich ist es ein sehr interessantes und zukunftsträchtiges Thema, was mich sehr motiviert hat während der Arbeit.
Das programmieren in C++ war zu Begin gewohnheitsbedürftig, ich habe zwar beide Module an der HSR besucht, jedoch gab es dort nur kleine spezifische Übungen. Das Modul hat sich vor allem auf spezifische C++ Konstrukte, welche im Standard sind, konzentriert. Die Studienarbeit hat mir sehr geholfen besseres und in einem grösseren Rahmen C++ zu schreiben.
Das Umschreiben von einer C/C++ eher funktionalen Library zu einer objektorientierter C++17-Standard Library war eine gute Erfahrung. Obschon gesagt wird, dass C eine Teilmenge von C++ ist wird einiges anders gelöst. Ab dem C++11-Standard sind bei C++ einige neue Konstrukte dazu gekommen, die es einfacher und eleganter machen ein Program umzusetzen.
Nicht alle im Team besuchen die gleichen Vorlesungen, was es etwas schwieriger gestaltet sich abzusprechen, wenn es Unklarheiten gibt. Im Software Engineering 1 Modul, wurde aufgezeigt, dass Team die nicht im gleichen Stockwerk arbeiten Mehraufwand haben. Deswegen ist es zukünftig sicher besser, wenn möglich, mehr Zeit für allfällige Rückfragen zu haben.
Die Arbeit in einem grösseren Softwareprojekt, im Vergleich zu kleinen Übungsaufgaben, war eine gute Erfahrung. Das designen von dem E-Voting-System war sehr spannend.
Vor dem Studium habe ich keine Ausbildung in Richtung Informatik abgeschlossen. Meine Programmiererfahrung beschränkt sich mehrheitlich auf den Inhalt des Studiums. Ich schätze es mit Teammitglieder mitarbeiten zu können, die viel Fachwissen haben. Ich kann viel lernen von meinen Teammitgliedern.

A.0.3 Romeo Spinas

Mir hat die Arbeit zum Thema Post-Quantum-E-Voting gefallen. Es war eine gute Gelegenheit, etwas mit Verschlüsselung zu machen. Anfangs war es recht schwer die Mathematik hinter der Verschlüsselung zu verstehen, das Paper dazu war anspruchsvoll. Mit Hilfe des C und C++ Code als Beispiel wurde es gegen Ende jedoch besser. Die Fehlersuche blieb jedoch bis zum Schluss anstrengend, da die Daten jeweils erst entschlüsselt werden mussten, bevor mit dem eigentlichen Debugging begonnen werden konnte. An das Programmieren in C++ habe ich mich schnell gewöhnt, obwohl der Unterschied zu den Aufgaben im Unterricht teils recht gross waren.
Die Projektplanung in der Gruppe hat aus meiner Sicht gut funktioniert, obwohl wir nicht dieselben Unterrichts-Module besucht haben. Dass alle Teammitglieder einen Tag freigehalten haben, um jede Woche zusammen den Stand der Arbeit zu besprechen, hat die Projektplanung stark vereinfacht. Für zukünftige Projekte kann man das sicher wieder so machen, da es die Kommunikation im Team allgemein vereinfacht. Das Projekt konnte nicht immer so einfach in Arbeitspakete aufgeteilt werden, da es zu Anfangs starke Abhängigkeiten zwischen diesen gab. Das Arbeiten an einem Projekt während des ganzen Semesters hat mir geholfen unter anderem im Bereich der Planung praktische Erfahrungen zu sammeln, was sonst wenig geübt werden kann. Die Zeitschätzung von Arbeitspaketen und das Design einer Softwarearchitektur aufgrund von definierten Anforderungen sind wichtige Aspekte die ich mit Hilfe dieses Projektes verbessern konnte.
Die gewonnenen Erfahrungen kann ich in der Bachelor-Arbeit und anderen zukünftigen Projekten gut gebrauchen.

B Eigenständigkeitserklärung

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Eigenständigkeitserklärung

Erklärung

Ich erkläre hiermit,

- dass ich die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt habe, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass ich sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben habe.
- das ich keine durch Copyright geschützten Materialien (z.B. Bilder) in dieser Arbeit in unerlaubter Weise genutzt habe.

Ort, Datum:
Rapperswil, 31.05.2019

L.Lätsch
Lukas Lätsch

R. Furrer
Rolf Furrer

R.Spinas
Romeo Spinas

C Projektplan

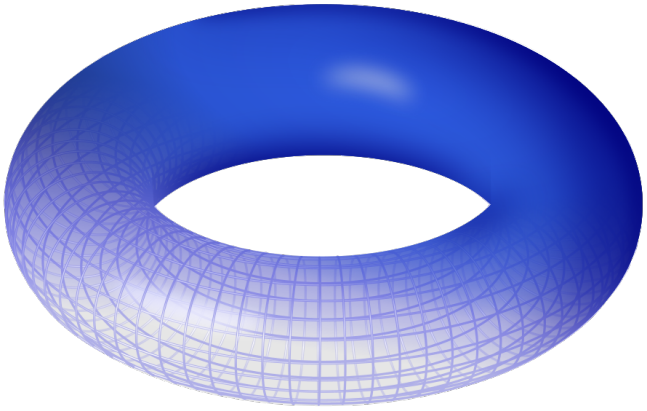
HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Post-Quantum E-Voting

Projektplan

Lukas Lätsch
Rolf Furrer
Romeo Spinas



Hochschule für Technik, Rapperswil, FS2019

Contents

1 Projekt Übersicht

Projektbeschreibung

Die Studienarbeit beinhaltet die Umsetzung eines Post-Quantum E-Voting. Die homomorphe Verschlüsselung und eine Beschreibung wie die für das E-Voting umgesetzt werden kann stützt sich dabei auf die Doktorarbeit *Towards efficient and secure Fully Homomorphic Encryption and cloud computing*[1].

Etwa 2 Jahre vor der Dokterarbeit sind die Grundlagen im Paper[2] vorgestellt worden. Dabei handelt sich es um eine erhebliche Beschleunigung des homomorphen Verfahren. Auf dieser Arbeit basiert haben die Autoren des Papers eine Library in C/C++ geschrieben geschrieben und auf Gihub `https://github.com/tfhe/tfhe` veröffentlicht.

Die wesentlichen Vorteile des Einsatzes der Verschlüsselungsmethode sind folgende:

- Die homomorphe Verschlüsselung wurde erheblich beschleunigt
- Die homomorphe Verschlüsselung erlaubt es auf verschlüsselten Daten logische Operationen auszuführen
- Die Verschlüsselungsmethode ist Quantum-Resistent

Basierend auf den oben erwähnten Papers und der auf Github Library soll eine E-Voting Applikation entwickelt werden. Dabei wird die Library erheblich angepasst, zum einen um sie ganze objektbasiert umzusetzen und zum anderen um sie zu erweitern um die Funktionen die für das E-Voting gebraucht werden.

Das E-Voting System wird dabei folgende drei Applikationen umfassen:

- Voter
- Bulletin Board
- Trustee

1.1 Zweck und Ziel

Sinn und Zweck

Es soll das gelernte Wissen vom Studium an einem grösseren Projekt umgesetzt werden. Die Software Engineeringmodule, Informationssicherheitsmodule und C++ werden in diesem Projekt eine gute Grundlage für diese Studienarbeit bieten.

Zielsetzung

- Referenzimplementation der homomorphen Verschlüsselung für E-Voting
- Anforderungsspezifikation erstellen um den Funktionsumfang (Scope) zu definieren
- OO-Analyse, OO-Design einsetzen
- Architektur definieren und dokumentieren
- Programmieren im Team (mit Versionskontrolle und Build Server)
- Lokal und auf dem Buildserver statische und dynamische Checks aufsetzen und in den Entwicklungsprozess einfliessen zu lassen
- Tests und Reviews (Unit Test, Systemtests) planen und durchführen

- Aufwandschätzung lernen (vorab schätzen, Zeit aufschreiben, Soll/Ist-Vergleich machen)
- Gute, adäquate Dokumentation erstellen (für grösseres Team und grösseres Projekt)

Persönliche Ziele

Für die Durchführung dieses Projektes haben wir uns folgende persönlichen Ziele vorgenommen:

- Mehr Erfahrung mit C++ aneignen
- Sich in eine komplexere Materie einzuarbeiten
- Ein besseres Verständnis für Software mit besonderen Sicherheitsanforderungen aneignen

2 Projektorganisation

2.1 Organisationsstruktur

Das Projektteam besteht aus drei Entwicklern.

- Rolf Furrer
- Lukas Lätsch
- Romeo Spinas

Verantwortlichkeiten werden pro Task in der Iterationsplanung zugeteilt.

2.1.1 Externe Schnittstellen

Betreuer:

- Prof. Dr. Andreas Steffen

Es werden regelmässig Sitzungen mit dem Betreuer abgehalten um den Stand der Arbeit zu besprechen.

3 Management Abläufe

3.1 Kostenvoranschlag

Zeitbudget
Die Studienarbeit hat am Montag 11.02.2019 begonnen und das Kickoff-Meeting mit dem Dozent hat am darauf folgenden Tag stattgefunden. Das Projekt soll gemäss der offiziellen Terminübersicht am 31.05.2019 fertiggestellt und die Dokumente hochgeladen werden.

Es stehen 15 Wochen Wochen zur Verfügung um die Studienarbeit anzugehen. Dabei arbeitet jedes Mitglied 240 Stunden an diesem Projekt. Dies gibt ein Zeitbudget für die 3 Mitglieder von 720 Stunden.

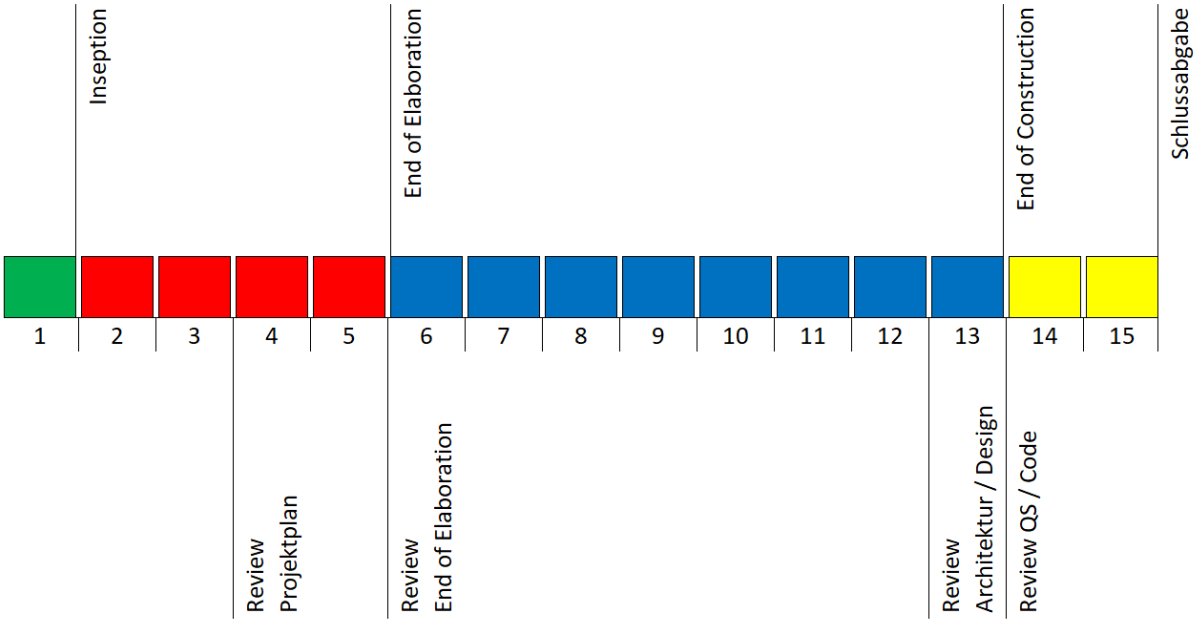
3.2 Zeitliche Planung

Zeiterfassung
Die Zeitplanung und die Verwaltung der Arbeitspakete wird die Software Redmine verwendet. Die Planung wird laufend aktualisiert, um schnell auf Änderungen reagieren zu können. Der aktuelle Zeitplan im Redmine ist unter folgender URL abrufbar: https://www.lag-13.ch/redmine/projects/studienarbeit/issues/gantt

Genauigkeit
Die effektive Zeit wird im Redmine auf 15 Minuten genau erfasst, um genug präzise den Ressourcenverbrauch des Projektes zu erfassen aber den Verwaltungsaufwand nicht unnötig in die Höhe zu treiben.

3.3 Phasen / Iterationen

Phasendauer
Die ersten vier Wochen des Projektes werden in einwöchige Iterationen aufgeteilt um sich in die Materie einzuarbeiten und um die Tool für den reibungslosen Ablauf bereitzustellen.



3.4 Meilensteine

Übersicht
Die Meilensteine, welche in der folgenden Tabelle ersichtlich sind werden angestrebt und nachfolgend mit dem verantwortlichen Dozenten besprochen.

Datum	Meilenstein	Beschreibung
10.03.2019	Projektplan abgeben	Projektplan erstellt/abgegeben
12.03.2019	Review Projektplan	
17.03.2019	Anforderungsspezifikation, Schnittstellen definiert	Use Cases erfasst, Nicht funktionale Anforderungen erfasst Beschreibung Schnittstellen zu anderen Systemen, Domainmodell erstellt, Architektur festgelegt, Abuse-Cases erarbeitet
24.03.2019	End of Elaboration	Prototype Objektorientierte Library
26.03.2019	Review End of Elaboration	
16.04.2019	Trustee Applikation	Prototype Trustee Applikation
30.04.2019	Voter Applikation	Prototype Voter Applikation
12.05.2019	Bulletin Board Applikation	Prototype Bulletin Board Applikation Kommunikation der 3 Applikationen funktioniert Feature Freeze
14.05.2019	Review Architektur / Design	
19.05.2019	End of Construction	Testing abgeschlossen
21.05.2019	Review QS / Code	
27.05.2019	Abstract	Abgabe Abstract and den Dozenten für ein Review freigeben
31.05.2019	Abgabe aller verlangten Dokumente	Hochladen aller verlagnten Dokumente auf archiv-i.hsr.ch Abgabe der Arbeit an den Dozenten bis 17:00 (Dozent Abgabe des Abstract)

Table 1: Meilensteine

3.5 Besprechung

Meetings
Die Projektmitglieder treffen sich mindestens einmal pro Woche. Die Meetings finden jeweils am Dienstag statt. Eine Standordsbestimmung findet jeweils vor dem Treffen mit dem Dozenten statt und danach wird die weitere Vorgehensweise zusammen bestimmt.

Reviews
Die Meetings mit dem betreuenden Dozenten finden wenn möglich am Dienstag um 13 Uhr statt, um ungeklärte Fragen zu besprechen und die nächsten Schritte zu planen. Zudem werden die Dokumente die bei den Meilensteinen abgegeben werden an diesen Meetings besprochen.

4 Risikomanagement

4.1 Risikomatrix

Nr	Titel	Beschreibung	Schaden pro Person [h]	Eintrittswahrscheinlichkeit	Gewichteter Schaden	Vorbeugung	Verhalten beim Eintreten
R1	Platform abfängigkeit	Die verwendeten Libraries können nicht Plattform unabhängig eingesetzt werden	8	Mittel	Mittel	Einrichte von Linux und Linuxänlichen Buildumgebungen (Cygwin)	Windows als Zilsystem ausschliessen
R2	Ressourcen	Die Laptops verfügen nicht über genügend Ressourcen um beispielsweise den Secretkey zu genereieren	16	Klein	Gross		Schlüssel auf einem Server generieren
R3	inkorrekte Wahlresultate	Fehler im Code oder manipulierter Input führt zu falschen Wahlresultaten	16	Mittel	Gross	Möglichst gründliche UnitTests mit Abusecases	
R4	Anonymität	Kommunikation über unsichere Kanäle kann zu Verlust der Anonymität von Wählern führen	4	Klein	Gross	Keine unverschlüsselte nachrihten versenden	

Table 2: Risikomatrix

5 Arbeitspakete

Die Arbeitspakete des Projektes werden mit Redmine verwaltet. Alle Arbeitspakete werden geschätzt und priorisiert und dann in eine Iteration eingeplant.

6 Infrastruktur

6.1 Verwendete Infra-struktur

In der nachfolgenden Liste befindet sich die für die Umsetzung des Projektes benötigte Hardware und Tools.

Name	Verwendung
Redmine	Planung
Gitlab	Versionsmanagement
Buildserver (Debian, Jenkins)	Deployment / Testing
Laptop (2 Linux, 1 Windows)	Developement
Cygwin	Linux änliche build Umgebung für Windows
Cmake	Zum generieren der Makefiles
Cevelop	C++ IDE

Table 3: Verwendete Infrastruktur

6.2 Verwendete Frameworks

Name	Verwendung
TFHE	Dient als grundlage, wird neu geschrieben
FFTW3	fast Fourier transform
GMP	Multiple Precision Arithmetic Library
Google Test	Unit Testing und Benchmarking
Cute	Unit Testing (Potenzieller ersatz für Google Test)

Table 4: Verwendete Frameworks

7 Qualitätsmassnahmen

Übersicht
Die folgende Tabelle gibt eine Übersicht über die festgelegten Qualitätsmassnahmen:

Massnahme	Zeitraum	Ziel
Backup aller Online-Tools	Einmal täglich	Kein Datenverlust, Konsistenz
Durchbesprechen der Dokumentation an Meetings	Jedes Meeting	Dokumentation enthält Konsens, keine Einzelmeinungen
Verwendung einer Projektmanagement-Software	Gesamtes Projekt	Zentralisierte Verwaltung von Pendenzen, Tickets, Zeiterfassung
Meilenstein-Snapshots	Jeder Meilenstein	Nachvollziehen des Ablaufs im Nachhinein möglich
Führen von Git-Repositories	Gesamtes Projekt	Versionierung, Sicherung
Implementation von Unit-Tests	Gesamtes Projekt	Korrektheit der Features
Durchführen von Code-Reviews	Nach Feature-Fertigstellung	Korrektheit und guter Stil im Code
Verwenden von Style-Guidelines	Gesamtes Projekt	Konsistenz im Code-Styling

Table 5: Qualitätsmassnahmen Übersicht

7.1 Dokumentation

Ablageorte
Die Dokumentationsdokumente der Post-Quantum E-Voting Arbeit, werden als LaTeX, in einem separaten GitLab Repository verwaltet.

Team-Konsens
Die Qualität der Dokumente wird in erster Linie dadurch sichergestellt, dass sämtliche Änderungen nachverfolgt und dann im jeweils nächsten Teammeeting (im Regelfall am nächsten Dienstagnachmittag) im Plenum besprochen und nachvollzogen werden. Auf diese Weise repräsentiert die Dokumentation immer den Konsens des gesamten Teams und nicht die Meinung von Einzelpersonen. Zudem wird das ganze Team über den aktuellen Stand des Projektes damit informiert. LaTeX Dokumente ermöglichen zusätzlich Änderungen einfach in einem Git-Diff nachzuverfolgen.

Snapshot
Bei Erreichen jedes Meilensteins wird ein Snapshot der gesamten Dokumentation als PDF erstellt und auf dem GitLab Repository hinterlegt, damit später der gesamte Prozessablauf noch nachvollzogen werden kann, sollte dies nötig werden.

7.2 Projektmanagement

Redmine
Das Projektmanagement wird wie im Modul "Software Engineering 1" vorgeschlagen das Tool Redmine eingesetzt, das auf einem privaten Server in einem Docker-Container läuft und somit für alle Mitarbeitenden jederzeit erreichbar ist. Das Tool ist unter der Adresse https://www.lag-13.ch/redmine/projects/studienarbeit

Verwendung
Redmine wird primär zum Erstellen von Arbeitspaketen, zum Nachvollziehen der bereits erledigten Aufgaben und bekannter Bugs und zum Festhalten der aufgewendeten Zeit verwendet.

Backup
Täglich werden Backups des Inhaltes von Redmine vorgenommen, damit im Stör- oder Verlustfall der Schaden auf ein Minimum reduziert wird. Das Backup wird auf Dropbox eines Mitarbeitenden abgelegt, ist also Off-site und somit nicht betroffen, falls es ein Problem mit den privaten Server geben sollte.

7.3 Entwicklung

Versionierung
Der Source Code von der Studienarbeit wird mittels dem Git-Versionskontrollsystem verwaltet und befindet sich auf dem offiziellen GitLab Servern. Architekturbedingt werden mehrere Git-Repositories erstellt, es wird für den Teil Voter-Applikation, Bulletin-Board-Applikation, Trustee-Applikation und die Sharred-Library für diese Projekte ein Repository erstellt um eine klarere Trennung zu ermöglichen.

Backup
Das Repository wird auf jedem Notebook der Teammitglieder und noch bedingt auf den privaten PC aktuell gehalten, sodass das mintestens drei Kopien der Repositories bestehen.

Projektautomation
Für die Builderstellung, Automatisierung, Testing und Continuous Integration (CI) wird das Tool Jenkins eingesetzt, das unter `https://jenkins.lag-13.ch` für alle Developer verfügbar ist. CMake (Cross-platform C++ Build Generator) wird zusammen mit Conan.io (Dependency Manager) verwendet um die CI umzusetzen. Verantwortlich für Jenkins ist ebenfalls Lukas Lätsch.

7.3.1 Vorgehen

Vorgehensmodell
Die Entwicklung erfolgt nach dem in der Vorlesung "Software Engineering 1" betrachteten Mischmodell aus RUP und Scrum, bekannt als Scrum+. Die Scrum-Meetings finden jeweils am Ende einer Iteration als Teil des Dienstagnachmittag-Meetings statt. Eine Iteration hat die Länge von zwei Wochen (mit einer Ausnahme, bedingt durch die 15 Wochen Projektlänge).

7.3.2 Unit-Testing

Vorgehen
Jeder Developer ist selbst dafür verantwortlich, dass sein Code von Unit Tests ausreichend abgedeckt wird und führt die Tests und Coverage-Metrics während dem Development auch selbst lokal aus. Im C++ wird hierzu GoogleTest und CUTE-Tests verwendet. Diejenigen Developer, die mit ihm am selben Modul arbeiten, reviewen diese Tests nach deren Fertigstellung und prüfen, ob eventuell noch Edge Cases nicht abgedeckt sind, Überlegungsfehler vorliegen und so weiter. Fehlgeschlagene Tests werden unter keinen Umständen ignoriert, sondern das Problem (sofern sinnvoll) behoben, bevor das nächste Feature angefangen wird.

Automation
Zusätzlich wird auf dem Jenkins-Server ebenfalls getestet, indem die gesamte erstellte Test-Suite (inkl. Integration Tests) dort ebenfalls ausgeführt werden kann. Dies geschieht jeweils, wenn gebuildet wird. Auch hier gilt: Kein Feature hat höhere Priorität als das Reparieren des Builds.

Verwendung
Für komplexe Features kann wahlweise das Prinzip des Test-Driven Development verwendet werden. Die

erwünschten Resultate eines neuen Features werden vor der Implementation in Mikrotests festgelegt, erst danach wird die tatsächliche Implementation erstellt.

Tools
Die Test-Coverage der in Cevelop und CLion entwickelten Module wird mit einem Tool überprüft.

Da die Software sehr leistungsintensive ist wird Google-Benchmark verwendet um die Performance zu messen. Zudem werden wird damit die Performance analysiert und wenn nötig optimiert.

7.3.3 Code Reviews

Zeitpunkt
Code Reviews erfolgen regelmässig nach Fertigstellung eines Features von ausreichender Komplexität mit mindestens einem anderen Developer, welcher am selben Modul mitarbeitet. Der Autor des Features erklärt seinen Code den anderen Beteiligten, welche ihm Feedback und gegebenenfalls Verbesserungsvorschläge geben. Auf jeden Fall erfolgt ein Review am Ende eines Sprints. Nach Bedarf kann auch zwischendurch ein Review angesetzt werden.

Unterstützung
Sollten im Verlauf des Development an einer Stelle Schwierigkeiten auftreten, so wird bevorzugt auf die Technik des Pair-Programming zurückgegriffen, wobei ebenfalls einer der Developer, die sich am gleichen Modul beteiligen, die Rolle des zweiten Augenpaares übernimmt.

Wenn spezifische unlösbare Probleme bestehen wird bei bedarf bei den Spezialisten an der Schule zurückgegriffen.

7.3.4 Code Style Guidelines

Die Richtlinien beinhalten folgende Punkte:

- Es soll der neuste Standard C++17 eingesetzt werden wenn sie die Möglichkeit bietet.
- Die C++ Core Guidlines sollen dort eingesetzt werden, wo sie Sinn machen und wo als gut geheissen worden sind aus dem Modul C++-Advanced
- Die gelernten Richtlinien vom Fach C++ und C++-Advanced sollen vorgezogen werden.

7.4 Testen

Unit Test
Unit Test werden wie oben unter dem Abschnitt 7.3.2 Unit Testing beschrieben durchgeführt.

Integration Test
Für Integration Testing wird Jenkins verwendet.

System Test
System Test werden von den Entwicklern jeweils am Ende einer Iteration anhand von Use Cases durchgeführt. Die Tests werden protokolliert.

References

[1] Ilaria Chillotti. *Towards efficient and secure Fully Homomorphic Encryption and cloud computing*. L'Université Paris-Sacla on Github. https://ilachill.github.io/papers/these_Ilaria_Chillotti_wo_acknowl.pdf. 2018.

[2] Ilaria Chillotti et al. *Faster Fully Homomorphic Encryption: Bootstrapping in less than 0.1 Seconds*. Cryptology ePrint Archive, Report 2016/870. https://eprint.iacr.org/2016/870. 2016.

List of Tables

D Anforderungsspezifikationen

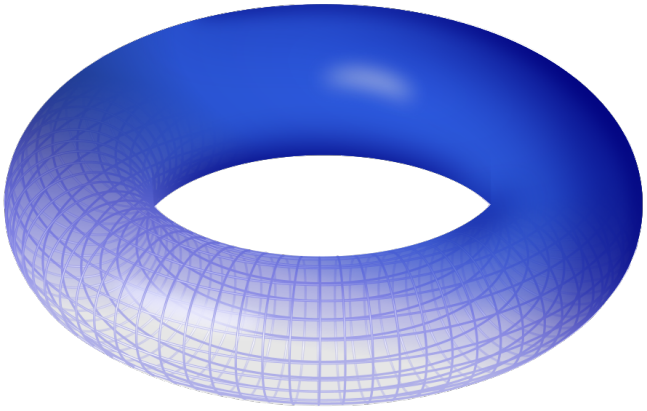
HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Post-Quantum E-Voting

Anforderungsspezifikation

Lukas Lätsch
Rolf Furrer
Romeo Spinas



Hochschule für Technik, Rapperswil, FS2019

Contents

1 Usecases

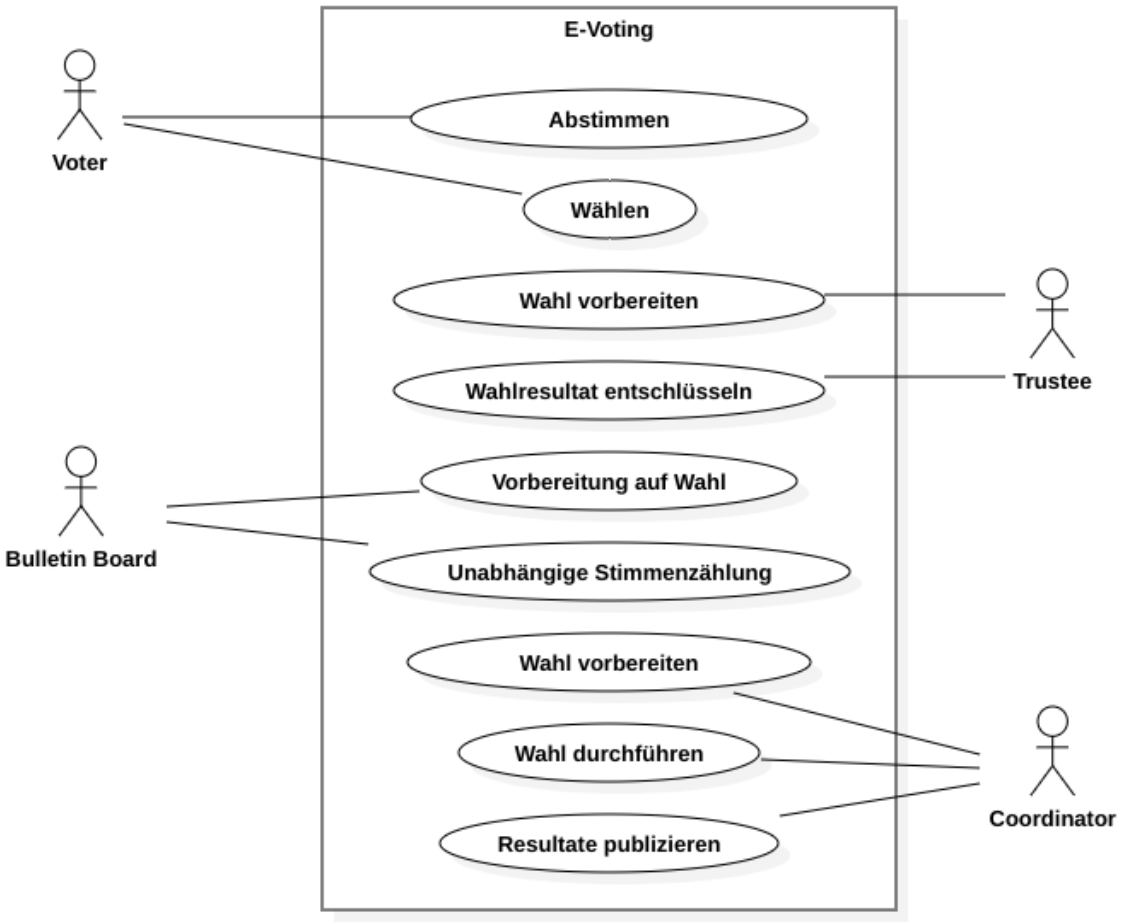


Figure 1: Usecases

1.1 Voter

Voter repräsentieren die Wähler unserer E-Voting Plattform.

Use Case ID:	1
Use Case Name:	Abstimmen
Actors:	Voter
Beschreibung:	Ein Wähler möchte an einer Abstimmung teilnehmen. Er kann sich zwischen Ja, Nein und Enthaltung entscheiden.
Preconditions:	Der Voter ist wahlberechtigt
Postconditions:	-
Normaler Ablauf:	1. Voter möchte seine Stimme abgeben 2. Coordinator liefert eine Liste mit Ja, Nein oder Stimmenthaltung und eine Public Key Liste. 3. Der Voter wählt eine der Möglichkeiten aus. 4. Der Voter schickt den ausgefüllten und verschlüsselten Wahlzettel ab.
Priorität:	Mittel
Frequency of Use:	Jeder Voter einmal, alle Stimmberechtigten
Business Rules:	Voter darf nur einmal eine Stimme abgeben

Table 1: Usecase: Abstimmen

Use Case ID:	2
Use Case Name:	Wahl
Actors:	Voter
Beschreibung:	Ein Voter möchte an einer Wahl teilnehmen mit einer Auswahl von Kandidaten
Preconditions:	Der Voter ist wahlberechtigt
Postconditions:	-
Normaler Ablauf:	1. Voter möchte seine Stimme abgeben 2. Coordinator liefert eine Liste mit allen Kandidaten und Stimmenthaltung. Und eine Public Key Liste. 3. Der Voter wählt eine der Kandidaten aus. 4. Der Voter schickt den ausgefüllten und verschlüsselten Wahlzettel ab.
Priorität:	Mittel
Frequency of Use:	Jeder Voter einmal, alle Stimmberechtigten
Business Rules:	Voter darf nur einmal eine Stimme abgeben

Table 2: Usecase: Wählen

1.2 Trustee

Trustees sind beispielsweise Parteien oder Kantone.

Use Case ID:	3
Use Case Name:	Trustee Wahl vorbereiten
Actors:	Trustee
Beschreibung:	Wahl soll von mehreren unabhängigen Trustees begleitet werden
Preconditions:	-
Postconditions:	Leere Stimmzettel sind erstellt
Normaler Ablauf:	1. Trustee erstellt geheimen Schlüssel. 2. Trustee generiert öffentliche Schlüssel. 3. Trustee stellt öffentliche Schlüssel dem Coordinator zu. 4. Trustee generiert KeySwitchingKey fürs Bulletinboard.
Priorität:	Mittel
Frequency of Use:	Für jeden Trustee einmal pro Wahl
Business Rules:	-

Table 3: Usecase: Wahl vorbereiten

Use Case ID:	4
Use Case Name:	Wahlresultat entschlüsseln
Actors:	Trustee
Beschreibung:	Wahlresultat soll von unabhängigen Trustees entschlüsselt werden
Preconditions:	Stimmen sind zusammengezählt
Postconditions:	-
Normaler Ablauf:	1. Jeder Trustee berechnet die jeweilige Teilentschlüsselung des Wahlergebnis.
Priorität:	Mittel
Frequency of Use:	Einmal pro Wahl
Business Rules:	-

Table 4: Usecase: Wahlresultat entschlüsseln

1.3 Bulletin Board

Das Bulletinboard ist vergleichbar mit einer Urne.

Use Case ID:	5
Use Case Name:	Bulletin Board Vorbereitung auf Wahl
Actors:	Bulletin Board
Beschreibung:	Wahl soll mit mehreren unabhängigen Bulletin Board begleitet werden
Preconditions:	-
Postconditions:	-
Normaler Ablauf:	1. Bulletin Board registriert sich beim Coordinator 2. Bulletin Board generiert den geheimen Schlüssel 3. Bulletin Board sendet den Public Key an die Trustees und erhält den KeySwitchKey zurück.
Priorität:	Mittel
Frequency of Use:	Einmal pro Bulletin Board
Business Rules:	-

Table 5: Usecase: Vorbereitung auf Wahl

Use Case ID:	6
Use Case Name:	Unabhängige Stimmenzählung
Actors:	Bulletin Board
Beschreibung:	Alle stimmen sollen unabhängig zusammengezählt werden
Preconditions:	-
Postconditions:	-
Normaler Ablauf:	1. Bulletin Board bekommt die Stimmzettel vom Voter. 2. Bulletin Board summiert die Stimmen verschlüsselt auf. 3. Bulletin Board postet das entschlüsselte Teilresultat und sendet es an den Coordinator.
Priorität:	Mittel
Frequency of Use:	Einmal pro Wahl
Business Rules:	-

Table 6: Usecase: Unabhängige Stimmenzählung

1.4 Coordinator

Der Coordinator ist verantwortlich für die Wahl.

Use Case ID:	7
Use Case Name:	Wahl vorbereiten
Actors:	Coordinator
Beschreibung:	Der Coordinator bereitet alles nötig für eine Wahl vor
Preconditions:	-
Postconditions:	Die notwendigen Schlüssel sind erstellt und die Kommunikation zwischen den Komponenten ist etabliert.
Normaler Ablauf:	1. Der Coordinator definiert die Trustees. 2. Der Coordinator definiert die Bulletin Boards. 3. Der Coordinator sammelt die Public Keys von den Trustees. 4. Der Coordinator erstellt ein Wahlpaket mit den Stimmzettel und den öffentlichen Schlüssel.
Priorität:	Mittel
Frequency of Use:	Einmal pro Wahl
Business Rules:	-

Table 7: Usecase: Wahl vorbereiten

Use Case ID:	8
Use Case Name:	Wahl durchführen
Actors:	Coordinator
Beschreibung:	Der Coordinator verteilt Stimmzettel.
Preconditions:	Wahlvorbereitungen sind abgeschlossen
Postconditions:	-
Normaler Ablauf:	1. Der Coordinator stellt den Voter die Stimmzettel zur Verfügung.
Priorität:	Mittel
Frequency of Use:	Einmal pro Wahl
Business Rules:	-

Table 8: Usecase: Wahl durchführen

Use Case ID:	9
Use Case Name:	Resultate publizieren
Actors:	Coordinator
Beschreibung:	-
Preconditions:	Die Wahl ist durchgeführt
Postconditions:	-
Normaler Ablauf:	1. Der Coordinator sammelt die zusammengezählten Stimmen. 2. Der Coordinator gibt die Stimmen den Bulletin Boards weiter. 3. Der Coordinator validiert das Resultat und veröffentlicht es.
Priorität:	Mittel
Frequency of Use:	Einmal pro Wahl
Business Rules:	-

Table 9: Usecase: Resultate publizieren

2 Abuse-Cases

Im Rahmen dieser Arbeit werden keine Signaturen verwendet für die Datenintegrität zu überprüfen, deswegen werden diese in den Abuse-Cases nicht aufgeführt.

2.1 Wahlverzögerung

Die Abstimmung kann durch Leute oder Organisationen, die dem Staat schaden zufügen wollen, politische Gegner, Script Kiddies oder durch andere verzögert werden. Dadurch kann es kurzfristig zu Aufruhr kommen und längerfristig zu politischen Instabilitäten.

2.1.1 Auslastung des Systems

Threat
Das System kann überlastet werden .Dadurch können Voter ihre Stimme nicht in einer gewissen Zeit (Voter frustriert) oder kurz vor dem Wahlende abgeben (Stimme nicht gezählt).

Massnahmen

- Die Voter auffordern, dass sie nicht zu spät ihre Stimme abgeben.
- Das Wahlende wenn nötig verschieben können.
- Mehrere Systeme (Bulletin Board) machen, die die Stimme entgegennehmen können.

2.1.2 Datenkorruption

Die Daten als Input dienen für eine Applikation können bewusst verändert werden.

Threat
Es könnten beliebige Daten generiert werden, die als Ballot an das Bulletin Board geschickt werden.

Massnahmen

- Es können nur registrierte Stimmbürger die Ballot absenden an das Bulletin Board.
- Der Parser muss so geschrieben werden, dass ungültige Daten schnell verworfen werden können.

Threat
Die Wahlzettel können so manipuliert werden, sodass das Bulletin Board abstürzt beim verarbeiten der Ballots.

Massnahmen

- Die Daten vorsichtig parsen, sodass nur valide Daten verarbeitet werden.

2.2 Wahlverfälschung

Wenn die Wahl durch Outsiders oder Insiders verändert werden kann.

2.2.1 Parallele Systeme zum Abstimmen

Es könnte sein, dass Dritte versuchen Einfluss zu nehmen auf das Wahlresultat.

Threat
Es könnten falsche Ballots ausgeteilt werden, welche dazu führen, dass die Voter einen invaliden Stimmzettel haben.

Massnahmen

- Die Ballots sollen signiert werden, sodass die Voter Applikation den leeren Stimmzettel verifizieren kann.

2.2.2 Erstellen einer neuen Wahl

Threat
Es könnte von Insidern neue Wahlzettel erstellt werden, ohne Zustimmung durch den Staat, und dann verteilt werden.

Massnahmen

- Das generieren von dem Secret-Key soll ähnlich durchgeführt werden wie der Root Zone KSK Rollover
- Das generieren der Keys findet vom Trustee offline statt. Die Stimmen werden dann mit einem Datenträger auf verschiedene Verteilsysteme gebracht, verifiziert und dann an die Voter on Request nach der Verifizierung verteilt. Für jeden Trustee müsste es 1 bis n Verteilsysteme haben.

2.3 Wahlzwang

2.3.1 Beeinflussung der Wähler

Die Voter können bestochen oder gezwungen werden eine bestimmte Wahl zu tätigen, durch die Familie oder durch Dritte.

Threat
Durch eine physische oder durch andere Drohungen kann zu einer bestimmten Abstimmung gezwungen werden.

Massnahmen

- Nach der Stimmabgabe kann die Abstimmung wieder verändert werden bis am Wahlende.
- Bedrohte Voter können sich an einer offiziellen Stelle melden innerhalb einer gewissen Frist nach Wahlende, um in einem zweiten Durchgang ihre Stimme abgeben. Dabei wird ihre erste Stimme invalidiert.

2.4 Falschaussagen

2.4.1 Entschlüsselung von Fake Daten

Threat
Es werden andere Resultate entschlüsselt die gar nicht der richtigen Wahl entsprechen.

Massnahmen

- Oversight bei der Entschlüsselung.
- Es sollen mehrere Trustees und Bulletin Board eingesetzt, die es braucht um die Daten erfolgreich zu entschlüsseln.
- Die Anzahl der abgegebenen Stimmen muss korrekt sein.

2.4.2 Nachvollziehbarkeit der Wahlresultate

Threat
Es können Falschaussagen über die Resultate getätigt werden. Unter Umständen die Zahlen völlig erfunden werden.

Massnahmen

- Wahlauswertung durch verschiedene Interessengruppen, wie zum Beispiel die Parteien und zusätzlich noch unabhängige Organisationen
- Wahlresultate nachvollziehbar und auch berechenbar machen, sodass jeder Stimmbürger mit genug technischen Fertigkeiten die Wahlresulate berechnen kann.

2.5 Hochrechnungen Stimmen

2.5.1 Trends der Abstimmung erkennen

Threat
Wenn die Resultate vor zu entschlüsselt werden können, können die Voter beeinflusst werden durch Hochrechnungen.

Massnahmen

- Einsatz von mehreren Trustees, sodass es alle braucht um die Stimmen Resultate entschlüsseln zu können.

3 Domainanalyse

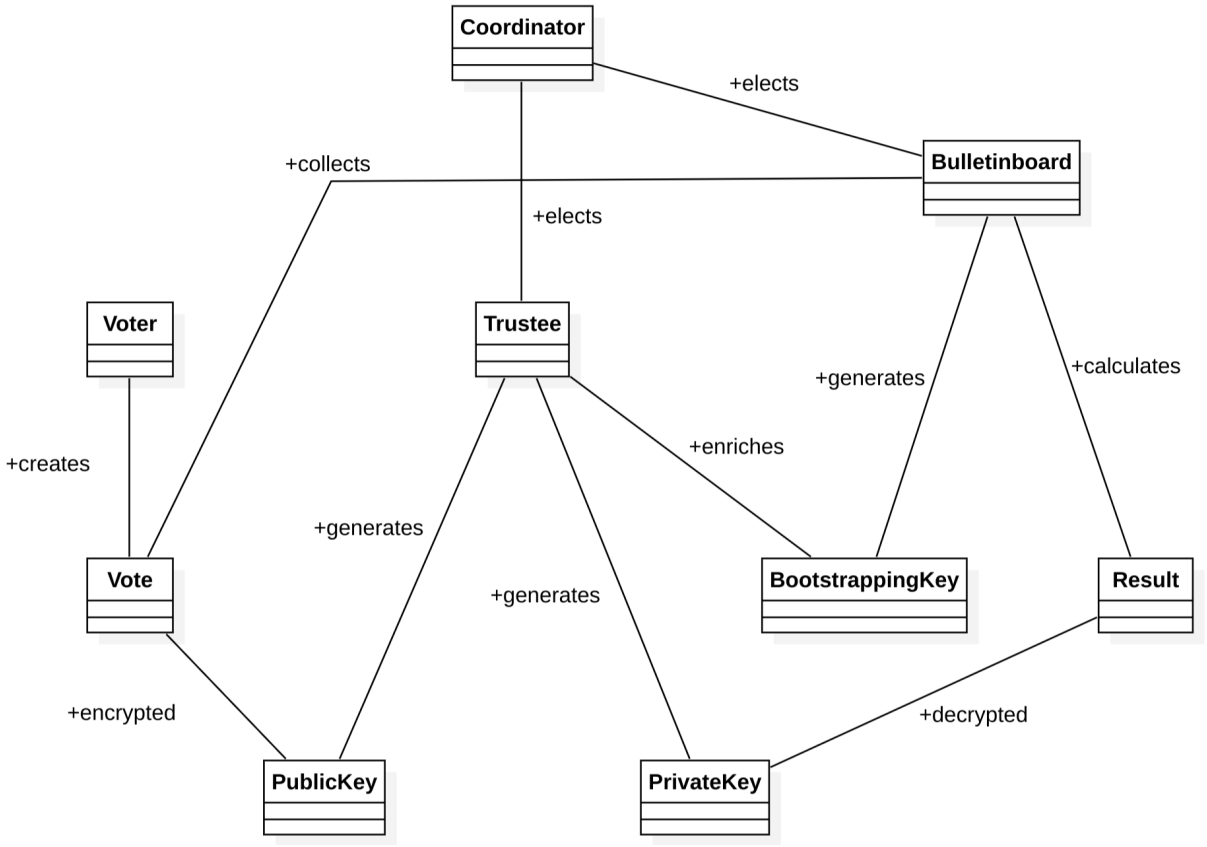


Figure 2: Domainmodel

3.1 Coordinator

Der Coordinator ist verantwortlich für die Wahl. Er definiert die Wahloptionen, die Trustees und die Bulletin Boards. Nach der Abstimmung veröffentlicht er die Wahlergebnisse.

3.2 Voter

Die Voter haben die Möglichkeit an einer Abstimmung teilzunehmen. Zu diesem Zweck füllen sie einen leeren Stimmzettel (Vote) aus und addieren dazu je einen öffentlichen Schlüssel jedes Trustees. Der Voter muss dabei mindestens einem Trustee vertrauen.

3.3 Trustee

Die Trustees generieren als Vorbereitung auf eine Abstimmung je einen privaten Schlüssel und viele öffentlichen Schlüssel. Zudem stellen alle Trustees gemeinsam dem Bulletinboard einen Key-Switch-Key zur Verfügung. Wenn eine Wahl abgeschlossen ist, erhalten alle Trustees die Wahlresultate, die sie nur gemeinsam entschlüsseln können.

3.4 Bulletinboard

Das Bulletinboard sammelt alle Wahlzettel und addiert sie auf. Es errechnet das Wahlresultat, kann es aber nicht selbst entschlüsseln.

4 Kommunikationsanalyse

Die verschiedenen Komponenten des Systems kommunizieren wie folgt.

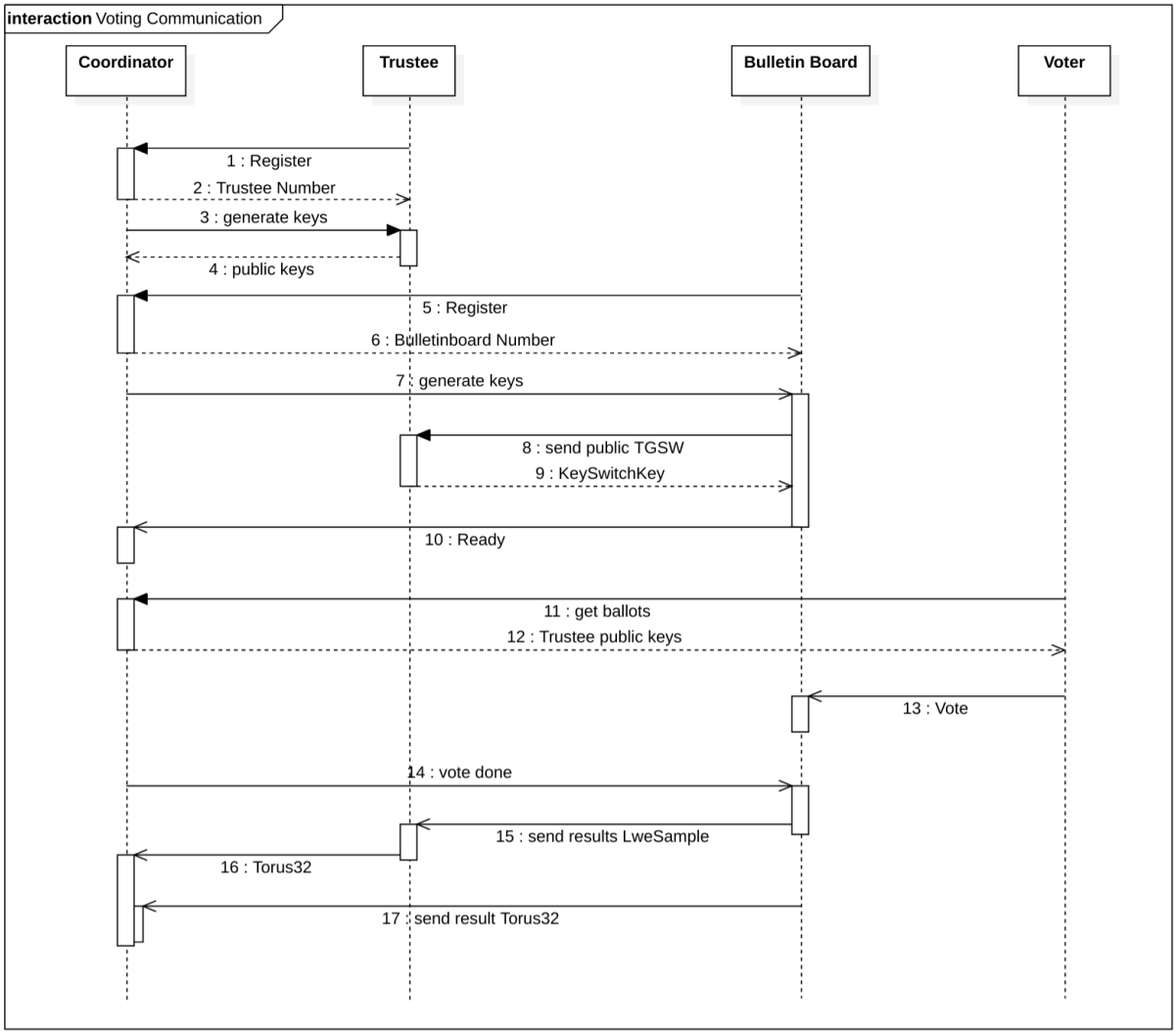


Figure 3: Kommunikationsanalyse

5 Nicht funktionale Anforderungen

Die nicht funktionalen Anforderungen sind begrenzt anwendbar in dieser Arbeit, da es vorerst um einen Working-Prototype geht. Das Ziel ist die Berechnung der Stimmen möglichst kurz zu halten.

5.1 Randbedingungen

Die mathematische Grundlage zur Umsetzung der Arbeit werden der Doktorarbeit [1] und der publizierten Library https://github.com/tfhe/tfhe entnommen.

5.2 Usability

Die Usability wird im Rahmen der Studienarbeit gering gehalten, es geht vorerst darum eine Proof-of-Concept umzusetzen des E-Voting-Systems.

5.3 Privacy

Der Secret-Key soll nach dem verwenden im RAM überschrieben werden und es darauf geschaut werden, dass das Memory, welches den Secret-Key enthalten nicht auf einem Datenträger abspeichert wird (swapping, paging).

Die Public Keys und die Ballot sollen zur Sicherheit auch nach Gebrauch aus dem Memory entfernt werden und nie geswapped.

5.4 Security

Die Schnittstellen sollen so ausgelegt werden, dass in einem Folgeprojekt die Daten sicher übertragen werden können.

Die Plattformsicherheit ist essentiell für alle Orte, wo eine der drei Applikationen darauf laufen. In dieser Arbeit wird darauf nur in einer architektonischer Sicht der Applikationen betrachtet. Die Gewährleistung der Sicherheit der Plattform auf welcher die Applikationen laufen sollte in einem Folgeprojekt beachtet werden.

5.5 Plattform

In erster Linie wird darauf geschaut, dass das Programme unter Linux laufen. In einem zweiten Schritt soll darauf geschaut werden, dass die Voter Applikation vor allem auch auf Windows und wenn genügend Zeit vorhanden ist sie auch auf OSX lauffähig ist.

Die Trustee und Bulletin Board Applikation muss nur auf Linux lauffähig sein, da die sie nicht für den Consumer gedacht sind. Es wäre natürlich wünschenswert, wenn das Bulletin Board auf Windows und OSX läuft, sodass Voter die Abstimmung auch auf ihrem eigenen Computer nachrechnen können.

Wenn die Capabilities weiterer Plattformen es unterstützen soll auch geschaut werden, dass diese unterstützt werden in diesem Projekt oder in einem Folgeprojekt.

5.6 Performance

Die Auswertung der Wahl sollte nicht länger als eine Woche dauern auf einem aktuellen Consumer-PC. Die Performance ist limitiert durch die gegebenen Algorithmen aus der Doktorarbeit. Die Performance in den drei zu erstellenden Applikationen kann optimiert werden, sie möglichst performant ablaufen. Das Program soll so optimiert werden, dass wenn möglich in einer Folgeprojekt Programmteile parallelisiert werden können.

Wenn die Zeit besteht soll 'Leveled mode LHE' implementiert werden, welche in der Doktorarbeit beschrieben ist aber in der vorhandenen Library noch nicht umgesetzt ist. Damit kann die Anzahl der binary Operationen bestimmt werden, sodass die Noise nicht zu gross wird und die Daten nicht unlesbar machen.

6 Einschränkungen

Wegen der zur Verfügung stehenden Zeit, wird der Umfang des Projektes um folgende Punkte limitiert:

- Keine Netzwerkkommunikation zwischen den Komponenten.
- Datenintegritäten werden nicht überprüft
- Nur der C/C++ Zufallszahlengenerator.
- Kein GUI.

References

[1] Ilaria Chillotti. *Towards efficient and secure Fully Homomorphic Encryption and cloud computing*. L'Université Paris-Sacla on Github. https://ilachill.github.io/papers/these_Ilaria_Chillotti_wo_acknowl.pdf. 2018.

Illustrationsverzeichnis

Tabellenverzeichnis

E Softwarearchtektur

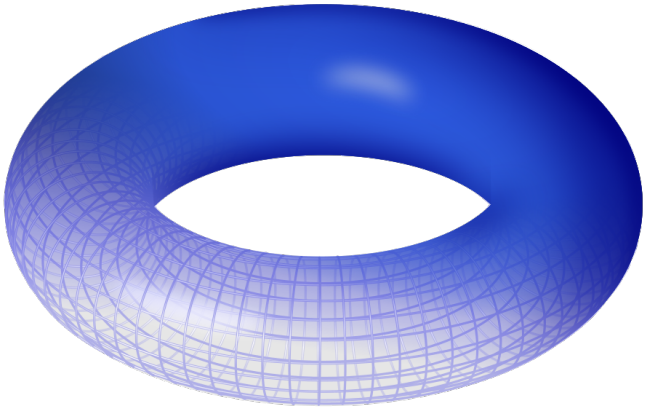
HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Post-Quantum E-Voting

Software Architektur

Lukas Lätsch
Rolf Furrer
Romeo Spinas



Hochschule für Technik, Rapperswil, FS2019

Contents

1 Systemübersicht

Die Applikation ist ein Proof-Of-Concept, deswegen werden die verschiedenen Komponenten in einem Prozess abgebildet. Die Komponenten sind so gegliedert und aufgebaut, sodass der Aufwand um die Komponenten auf verschiedenen Systemen zu verteilen gering ist. In der Systemübersicht unten ist auch die vereinfachte Interaktion und Reihenfolge der Komponenten aufgeführt.

Prozess

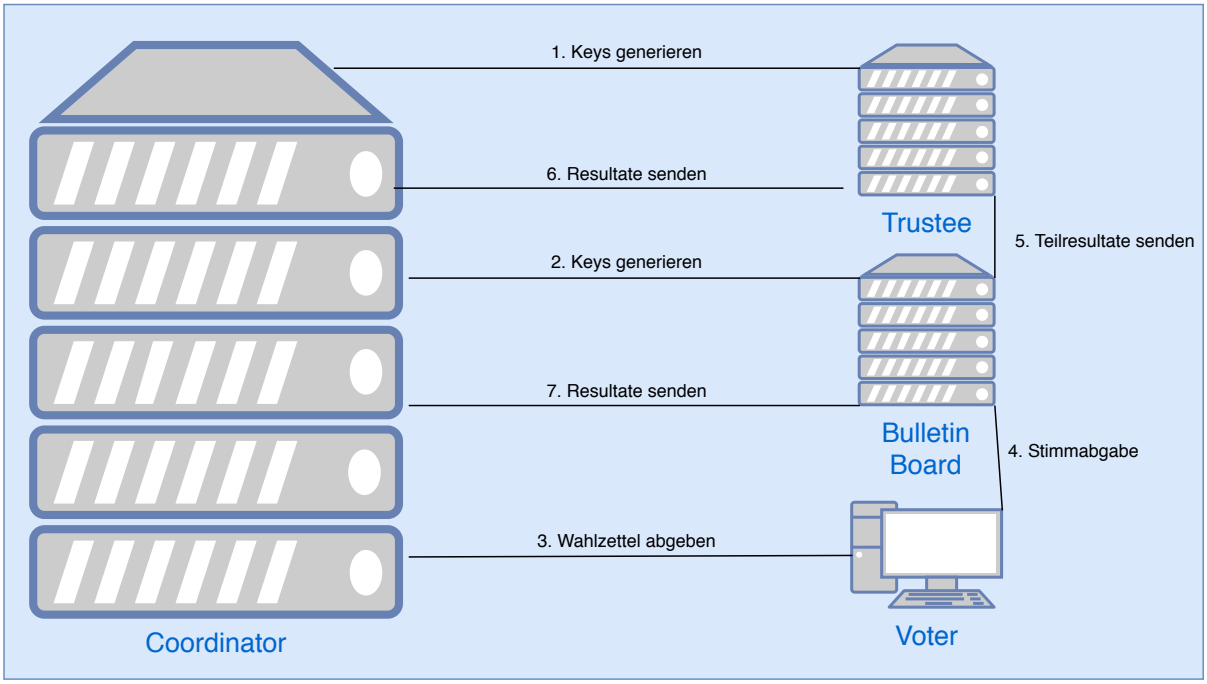


Figure 1: Systemübersicht

1.1 Coordinator

Der Coordinator organisiert die Interaktion zwischen den verschiedenen Systemen. Der Trustees und Bulletin Boards melden sich beim Coordinator. Danach werden die Aufträge von ihm verteilt. Der Coordinator ist nicht immer die Zwischenstelle, wie beim Punkt 5 und 4, sodass der Coordinator seine Position nicht missbrauchen kann.

1.2 Trustee

Die 3-n Trustees generieren, auf Anfrage vom Coordinator, die geheimen und öffentlichen Schlüssel und übergeben die öffentlichen dem Coordinator. Zudem generieren die Trustees den KeySwitchingKey für das Bulletin Board.

Wenn die Wahl abgeschlossen ist senden die Bulletin Boards den Trustees in der korrekten Reihenfolge die verschlüsselten Wahlergebnisse. Die Trustees Teilentschlüsseln ihr Teil und geben das Resultat nachfolgenden Trustee weiter.

1.3 Bulletin Board

Die 3-n Bulletin Boards generieren ihren öffentlichen und geheimen Schlüssel. Der öffentliche Schlüssel wird an die Trustees geschickt und daraufhin bekommt das Bulletin Board den KeySwitchingKey zurück.

Die Voter übergeben ihre Stimmzettel den Bulletin Boards. Die Bulletin Boards summieren homomorph die Stimmen auf. Wenn alle Stimmen abgegeben sind veröffentlichen die Bulletin Boards die Teilresulate und senden sie an den Coordinator.

1.4 Voter

Der Voter bekommt die Wahlzettel vom Coordinator. Er füllt den Wahlzettel aus und leitet sie an die Bulletin Boards weiter.

2 Architektonische Ziele und Einschränkungen

2.1 Ziele

Das Ziel der Arbeit ist es einen Working-Prototype eines Post-Quantum E-Voting System auf Grundlage von der Doktorarbeit 'Towards efficient and secure Fully Homomorphic Encryption and cloud computing'[1] zu erstellen.
Die folgenden Punkte werden besonders in dieser Semesterarbeit betrachtet:

- Die C/C++ wird umgeschrieben in eine objektorientierte C++ Library.
- Architektonische wird auf die Sicherheit geachtet (siehe Abuse-Cases im Dokument Anforderungsspezifikation).

2.2 Einschränkungen

Da es sich im ein Proof-Of-Concept handelt wird das E-Voting System minimal umgesetzt. Das heisst, dass folgende Limitationen gelten.

- Alle Komponenten des E-Voting Systems werden in einem Prozess angesiedelt. Es werden Schnittstellen verwendet, welche es später ermöglichen sollen das System ohne allzu grossen Aufwand netzwerkfähig zu machen (die Security ist dabei nicht beachtet).

3 Logische Architektur

3.1 Externe Schnittstellen

Die Applikation benötigt zum Funktionieren Zufallszahlen (Entropie). Alle Komponenten sind auf Entropie angewiesen, die jedoch in der Applikation selbst nicht oder nur sehr aufwendig bereitgestellt werden kann. Deshalb wird eine externe (ausserhalb der Applikation) Entropiequelle benötigt.

3.2 Architekturdiagramm

Für die Umsetzung des E-Voting Systems haben wir 4 Komponenten vorgesehen. Einerseits gibt es die vier Komponenten Coordinator, Voter, Trustee und Bulletinboard, die auch unterschiedliche Teilapplikationen darstellen. Zusätzlich gibt es die Core-Bibliothek, die die gemeinsamen Nenner aller Teilapplikationen implementiert.

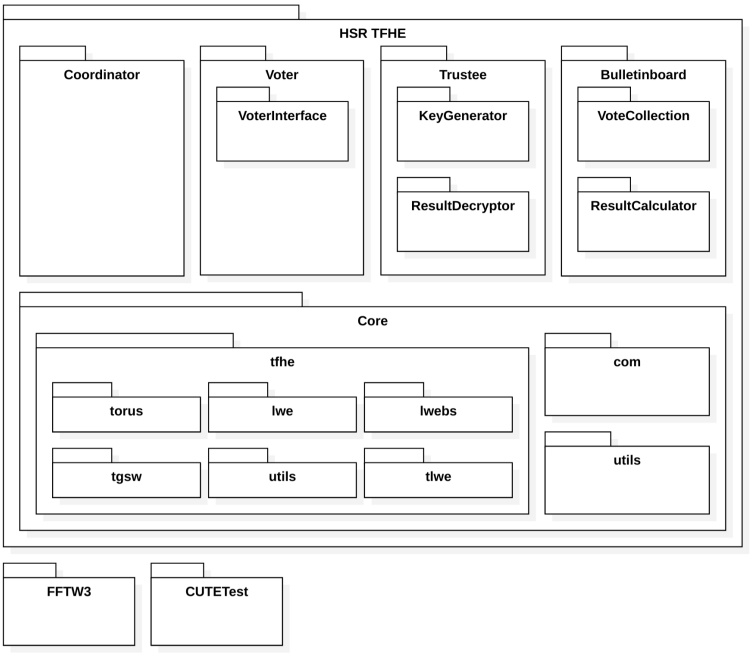


Figure 2: Architekturdiagramm

3.2.1 Core

Die Core-Bibliothek stellt alle Interfaces, die zur Kommunikation benötigt werden zur Verfügung. Im Rahmen dieses Projektes wird die Netzwerkkommunikation aber lediglich mit über Pure Virtual Functions abstrahiert. Später kann die Netzwerkkommunikation mithilfe von RPC umgesetzt werden. Zusätzlich wird die TFHE-Verschlüsselung in der Core-Bibliothek implementiert werden.

3.2.2 Voter

Die Voter Komponente ist die Benutzeroberfläche für die Wähler. Im Rahmen dieses Projektes wird es aber voraussichtlich lediglich eine Konsolenapplikation sein.

3.2.3 Trustee

Der Trustee ist für die Schlüsselgenerierung verantwortlich. Zudem hält sie sich bereit, den Voters die öffentlichen Schlüssel zur Verfügung zu stellen. Am Ende der Abstimmung entschlüsselt zudem jeder Trustee seinen Teil der Verschlüsselung.

3.2.4 Bulletin Board

Das Buletin Board sammelt die Stimmen und addiert diese auf.

4 Deployment

Die Applikation wird in einem Prozess ausgeführt, deswegen ist die Auslieferung relativ einfach. Die Applikation kann vom Jenkins in eine Zip-Datei verpackt werden und diese Datei kann dann heruntergeladen und ausgeführt werden. Wie in den nicht funktionalen Anforderungen erwähnt wird eine Linux Version priorisiert über einer Windows Version.

5 Datenspeicherung

Das Proof-Of-Concept braucht in der definierten Form keine Datenbank. Die generierten Daten werden auf dem Filesystem abgelegt. Dazu gehöhren die verschiedenen Schlüssel die zu einem späteren Zeitpunkt wiederverwendet werden müssen.

5.1 Abstimmungsdaten

Das Publizieren der Rohdaten, welche für die Nachprüfung gebraucht werden können, und die Wahlergebnisse werden in einem Konsolenprogramm angezeigt und wenn genügend Zeit vorhanden ist auch in einem Format abgespeichert welches in einer späteren Arbeit weiterverwendet werden kann.

6 Leistung

6.1 Standard Rechner

In der Entwicklungsphase werden die Computer der Studenten verwendet um die Berechnungen durchzuführen. Basierend auf der Doktorarbeit sollten sich die Berechnungszeiten erheblich schneller sein als vorherige Implementationen.

6.2 Institutsrechner

Wenn genügend Zeit vorhanden ist und der Institutsrechner frei ist, gibt es die Möglichkeit dort die Berechnungen auszuführen. Dies wäre interessant um einen Vergleich zu bekommen.

Illustrationsverzeichnis

F Software Lizenz

F.1 TFHE

Die Library auf der das E-Voting-System aufgebaut ist, basiert auf der veröffentlichten TFHE Library die unter https://github.com/tfhe/tfhe erhältlich ist. Die Library hat eine Apache License, Version 2.0.

F.2 Boost

Die Boost-Library wird im E-Voting-System verwendet, diese hat die Boost Software License. Mehr Informationen können auf der Webseite https://www.boost.org/ nachgeschlagen werden.

F.3 FFTW3

Die FFTW3 Library, die in der ursprünglichen Library und im E-Voting-System verwendet wird, ist unter der GPL lizenziert. Die offizielle Webseite ist unter http://www.fftw.org/ erreichbar.

F.4 E-Voting-System

Unter Beachtung der oben genannten Lizenzen wird die das E-Voting-System unter die Apache License, Version 2.0 gesetzt.

F.5 Urheberrecht und Nutzungsrechte

Die Urheber- und Nutzungsrechte sind auf der folgenden Seite angefügt.

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Vereinbarung über Urheber- und Nutzungsrechte

1. Gegenstand der Vereinbarung

Mit dieser Vereinbarung werden die Rechte über die Verwendung und die Weiterentwicklung der Ergebnisse der Studienarbeit Post-Quantum E-Voting' von Lukas Lätsch, Rolf Furrer, Romeo Spinas unter der Betreuung von Prof. Dr. Andreas Steffen geregelt.

2. Urheberrecht

Die Urheberrechte stehen der Studentin / dem Student zu.

3. Verwendung

Die Ergebnisse der Arbeit dürfen sowohl von der Studentin / dem Student wie von der HSR nach Abschluss der Arbeit verwendet und weiter entwickelt werden

Rapperswil, den 31.05.2019

L. Lätsch
Die Studentin/der Student

R. Furrer
Die Studentin/der Student

R. Spinas
Die Studentin/der Student

Rapperswil, den.....

.....
Der Betreuer der Studienarbeit

G Bedienungsanleitung Software

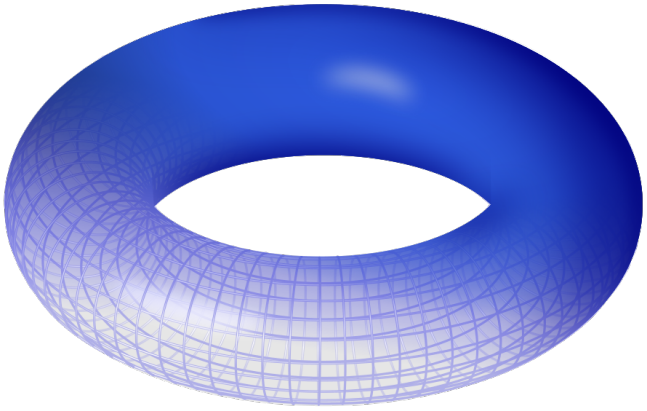
HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Post-Quantum E-Voting

Bedienungsanleitung Software

Lukas Lätsch
Rolf Furrer
Romeo Spinas



Hochschule für Technik, Rapperswil, FS2019

Contents

1 Kompilierung

1.1 Voraussetzungen

Die Voraussetzungen das E-Voting-System zu kompilieren sind wie folgt:

- CMake
- C++17 compiler
  - Linux: GCC/G++ 8.2+
  - Windows: MinGW oder WSL
- Libraries
  - Boost (1.69.0 used in testing)
  - FFTW3 (3.3.5 used in testing)

Nach dem Klonen des proof-of-concept Repository muss darauf geachtet werden, dass alle Git-Submodules geladen werden.

1.2 Setup

Am einfachsten kann man die E-Voting-System Applikation mit CMake builden. Dazu muss CMake auf dem System installiert sein. Die offizielle Webseite von CMake ist `https://cmake.org/`. Dort kann für Windows das Setup heruntergeladen werden. Für Linux ist CMake meist über den Package-Manager installierbar.

Die folgenden Kommandos müssen in der Konsole ausgeführt werden im proof-of-concept Ordner:

Linux

```
mkdir build
cd build
cmake ..
make
```

Windows

Unter Windows muss die FFTW3 Library in den proof-of-concept Ordner extrahiert werden.

```
mkdir build
cd build
cmake ../ -G "MinGW Makefiles";
make
```

1.3 Tests

Um das Core-Projekt zu testen muss die Datei fftw3.dll oder fftw3.so in die PATH Variable eingetragen werden. Es ist alternativ auch möglich die Library in das selbe Verzeichnis zu kopieren, in der das Core-Executable residiert.

2 Software Anleitung

2.1 Applikation starten

Nach dem kompilieren des 'Proof of Concept', was im Kapitel 1 beschrieben wird, kann von selben Verzeichnis aus das E-Voting-System aus der Konsole wie folgt gestartet werden:

```
./build/coordinator/exec/coordinatorexec
```

2.2 Applikation bedienen

2.2.1 Hauptmenü

Nach dem Starten der E-Voting-System Applikation wird folgendes textbasierte Menü angezeigt. Es kann die Anzahl der Trustees, Bulletin Board und Voters angepasst werden. Zudem können im Untermenü die Ballot Bits und die Voting Choices bestimmt werden.

```
************************** Main Menu *********************************

************************************************************************
 1 - Number of trustees to create           (1)  [ default value ] )
 2 - Number of bulletin boards to create    (1)  [ default value ] )
 3 - Number of voters boards to create      (1)  [ default value ] )
 4 - Set Parameters.
 5 - Get started with the creation of the entities.

 9 - Exit program
************************************************************************

 Enter your choice and press return:
```

Links wird die Nummer dargestellt, die für das Erreichen des Option gewählt werden muss. Es sind bereits Default-Werte festgesetzt, die bei bedarf angepasst werden können. Wenn man die Nummer eingibt und sie mit der Enter-Taste bestätigt kommt man zur gewünschten Option.

2.2.2 Untermenü

Das Untermenu kann mit Eingabe von 4 im Hauptmenü und mit Bestätigung von der Enter-Taste angezeigt werden. Wenn das Untermenü ausgewählt wurde wird folgendes Menü angezeigt:

```
************************** Sub Menu **********************************

************************************************************************
 1 - Number of ballot bits                  (7)  [ default value ] )
 2 - Number of voting choices               (3)  [ default value ] )
 3 - Back to main menu.
************************************************************************

 Enter your choice and press return:
```

Die Ballot Bit unter der Position bestimmt wie viele mögliche Bit zur Verfügung stehen um aufsummierte Stimmen abzuspeichern. Die Voting Choices bestimmen wie viele Wahlmöglichkeiten bestehen. Mit den gesetzten Default-Werten ist es möglich maximal $2^7 = 128$ Stimmen zu Zählen. Wenn die Voting Choices den Default-Werte hat, gibt es $2^3 = 8$ Wahlmöglichkeiten.

Wenn die Option 3 gewählt wird kommt man wieder ins Hauptmenü zurück.