Institut für Informationssicherheit

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit

Erweiterung von Ordinos um politische Wahlverfahren: House of Commons, Storting (Norwegen), Deutscher Bundestag

Carmen Wabartha

Studiengang:	Informatik
Prüfer/in:	Prof. Dr. Ralf Küsters
Betreuer/in:	Julian Liedtke, M.Sc.
Beginn am:	20. Oktober 2021
Beendet am:	20. April 2022

Kurzfassung

Damit E-Voting Systeme in der Praxis eingesetzt werden können, müssen sie eine Vielzahl an Eigenschaften erfüllen. So sollen sie nicht nur das Ergebnis einer Wahl berechnen, sondern auch funktionieren, wenn ein Teil derer, die an der Auswertung beteiligt sind, unehrlich sind. Auch in diesem nicht optimalen Fall muss sichergestellt werden, dass das Wahlgeheimnis gewährleistet bleibt, dass jeder Wähler überprüfen kann, ob seine Stimme gezählt wurde, dass das Wahlergebnis korrekt ist und dass Teilnehmer, die die Berechnung manipulieren, zur Verantwortung gezogen werden können.

Eine besondere Eigenschaft von sicheren E-Voting Systemen ist das Tally Hiding. Dabei bleiben einzelne Wählerstimmen und Zwischenergebnisse der Wahl wie beispielsweise die Anzahl der Stimmen pro Partei verschlüsselt und nur das benötigte Ergebnis wie die finale Sitzverteilung wird entschlüsselt. Tally Hiding bewirkt, dass die Vote Privacy deutlich erhöht wird und einige Angriffe nicht mehr möglich sind [31].

Ordinos ist ein E-Voting System, das sowohl Tally Hiding als auch Vote Privacy und Accountability sicherstellt [31]. Das Grundgerüst von Ordinos wurde bereits entwickelt und Evaluierungsfunktionen für einige Wahlen sind ebenfalls implementiert [25]. In dieser Arbeit wird Ordinos um einige Parlamentswahlen, wie sie in der Realität durchgeführt werden, erweitert. Konkret werden die Wahlverfahren für das britische House of Commons, das norwegische Storting und den Deutschen Bundestag hinzugefügt. Im Gegensatz zu vorherigen Wahlverfahren liegt nun der Fokus darauf, die Wahlen so umzusetzen, wie sie in der Praxis durchgeführt werden. Daher wird nicht mehr von der konkreten Wahl abstrahiert, sondern es müssen alle Feinheiten der Wahlen beachtet werden.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Algorithmen

1 Einleitung

Die zunehmende Digitalisierung unseres Alltags hat sich auch auf Wahlprozesse ausgedehnt. So gibt es immer mehr Anbieter, die E-Voting Systeme zur Online-Abstimmung zur Verfügung stellen und dabei ein vielfältiges Angebot haben [1], [2]. Zielgruppe dieser Dienste sind jedoch meistens private Kunden, Unternehmen oder ähnliche Organisationen, seltener große politische Wahlen. Grund hierfür ist vor allem das fehlende Vertrauen in E-Voting Systeme, da der gesamte Prozess weniger transparent scheint und daher Sorge vor Manipulationen des Ergebnisses oder Aufhebung des Wahlgeheimnisses besteht [3], [38], [14]. Vor allem bei großen politischen Wahlen herrschen daher oft Zweifel, ob E-Voting Systeme den Grundsätzen einer demokratischen Wahl gerecht werden können.

Das E-Voting System Ordinos ist durch mehrere Sicherheitseigenschaften so konzipiert, dass Manipulationen nicht unbemerkt durchgeführt werden können, das Wahlgeheimnis erhalten bleibt und keine unnötigen Informationen preisgegeben werden [31]. Besonders hervorzuheben ist die Eigenschaft Tally Hiding. Das bedeutet, dass die Full Tally, die einzelne Wählerstimmen und alle Zwischenergebnisse enthält, verschlüsselt bleibt, und nur das Wahlergebnis entschlüsselt wird. Dass keine unnötigen Informationen entschlüsselt werden, hat mehrere Vorteile: Die Vote Privacy wird erhöht, ein Gewinner einer Wahl wird eher als legitim angesehen und langfristigen Manipulationen wie das Verschieben von Wahlkreisgrenzen wird vorgebeugt. Auch einige Angriffe auf das Wahlprotokoll, die sich zusätzliche Informationen der Wahl neben dem eigentlichen Ergebnis zunutze machen, können verhindert werden [31].

Ordinos wurde bereits zu großen Teilen implementiert, sodass Wahlen mit generischen Verfahren wie Borda oder Condorcet durchgeführt werden können [24]. Diese Wahlverfahren sind allgemein gehalten und lassen sich auf viele Wahlen übertragen, abstrahieren dadurch aber von den Details und Feinheiten der tatsächlich durchgeführten Wahlen. In dieser Bachelorarbeit wird Ordinos um politische Wahlverfahren erweitert, wie sie in der Praxis durchgeführt werden, und es wird untersucht, wie gut sich Ordinos für solche Wahlverfahren eignet. Dabei müssen alle Feinheiten der Wahlen beachtet werden wie beispielsweise 5%-Hürden und Direktmandate statt nur einer klassischen parlamentarischen Verteilung. Im Zuge dieser Arbeit werden die Wahlen für das britische House of Commons, das norwegische Storting und den Deutschen Bundestag untersucht und in Ordinos umgesetzt. Dazu werden verschiedene Evaluierungsprotokolle entwickelt. Diese erhalten als Input die verschlüsselten Stimmzettel und generieren daraus das Endergebnis der Wahl.

Die Herausforderung dieser Arbeit liegt vor allem darin, dass die Evaluierungsprotokolle für die Praxis entwickelt werden sollen. Das bedeutet einerseits, dass alle Details und Sonderregelungen vollständig berücksichtigt werden müssen, sodass die Auswertung immer zum richtigen Ergebnis kommt. Dabei müssen auch Randfälle wie der Gleichstand zweier Kandidaten beachtet werden, auch wenn diese bei großen politischen Wahlen nur mit geringer Wahrscheinlichkeit vorkommen. Andererseits muss die Implementierung so optimiert werden, dass die Auswertung einer Wahl möglichst effizient ist. Da alle Rechenschritte verschlüsselt ausgeführt werden müssen, benötigen

auch einfache Vergleichsoperationen einige Sekunden Zeit für die Berechnung. Dies führt dazu, dass eine direkte Umsetzung der Wahlverfahren meist nicht möglich ist, sondern stattdessen andere Algorithmen entwickelt werden müssen, die für Ordinos optimiert sind. Zusätzlich muss jede Auswertung einer Wahl beweisbar sicher in Bezug auf die Anforderungen von Ordinos sein, das heißt, Tally Hiding, Vote Privacy und Accountability müssen gewährleistet werden.

Diese schriftliche Dokumentation der Arbeit ist dabei folgendermaßen aufgebaut: Zuerst beschreibt Kapitel 2 die Grundlagen für das bereits existierende E-Voting System Ordinos. Dabei wird der Aufbau von Ordinos skizziert, die Sicherheitseigenschaften definiert sowie die vorhandenen arithmetischen und logischen Grundbausteine vorgestellt. Anschließend wird in Kapitel 3 der aktuelle Stand von Ordinos beschrieben und gezeigt, wo diese Arbeit anschließt. Kapitel 4 enthält eine Literaturrecherche verschiedener in der Realität angewandter Wahlverfahren, von denen drei in den folgenden Kapiteln umgesetzt werden. Daraufhin wird in Kapitel 5 gezeigt, welche allgemeinen Änderungen beim Übergang von generischen Wahlen zu realen Wahlen zu beachten sind. Diese Änderungen sind für fast alle der umgesetzten Wahlverfahren relevant. Den Hauptteil der Arbeit bildet Kapitel 6. Hier wird an drei Beispielen konkret gezeigt, wie ein politisches Wahlverfahren aus der Praxis in Ordinos umgesetzt werden kann. Dazu werden die jeweils entwickelten Algorithmen vorgestellt und die Implementierung wird dokumentiert. Zusätzlich wird nachgewiesen, dass die vorgestellten Evaluierungsfunktionen korrekt sind, die Sicherheitseigenschaften von Ordinos erfüllen und effizient genug für reale Anwendungen sind. Als letztes werden die Ergebnisse in Kapitel 7 zusammengefasst, die Grenzen der Arbeit aufgezeigt und ein Ausblick für zukünftige Arbeiten gegeben.

2 Ordinos

Dieses Kapitel beschreibt das bereits existierende E-Voting System Ordinos [31], [25], das in der vorliegenden Arbeit um einige Wahlverfahren erweitert wird. Zuerst werden die wichtigsten Komponenten, aus denen Ordinos aufgebaut ist, beschrieben, sowie gezeigt, aus welchen Phasen sich die Durchführung einer Wahl mit Ordinos zusammensetzt. Im nächsten Teilkapitel werden die Sicherheitseigenschaften gezeigt, die Ordinos garantiert. Hier ist insbesondere das Tally Hiding hervorzuheben, für das Ordinos bekannt ist. Anschließend werden einige der schon implementierten arithmetischen und logischen Grundbausteine beschrieben, die im Zuge dieser Arbeit relevant werden.

2.1 Aufbau von Ordinos

Ordinos besteht aus einer Menge an Wählern, dem Bulletin Board, dem Authentication Server, der Election Authority sowie mehreren Trustees [31]. In den folgenden Absätzen werden diese mit ihren wichtigsten Funktionen vorgestellt und im nächsten Abschnitt wird gezeigt, wie die einzelnen Instanzen sich bei einer Wahl verhalten. Es wird davon ausgegangen, dass alle Komponenten für ihre Kommunikation authentische Kanäle nutzen.

Die *Wähler* lassen sich in zwei Mengen einteilen, die ehrlichen und die unehrlichen Wähler. Beide können ihre Stimmzettel für die Wahl abgeben, aber die ehrlichen Wähler wählen geheim nach einer Verteilungsfunktion und geben gültige Stimmen ab, während die unehrlichen Wähler vom Angreifer kontrolliert werden. Dieser entscheidet, ob und welche Partei(en) die unehrlichen Wähler wählen [31]. Außerdem besitzt jeder Wähler ein Voter Verification Device, das sich korrekt verhält.

Das *Bulletin Board* speichert alle Informationen, die für eine Wahl und deren Auswertung benötigt werden. Dies sind insbesondere alle Konfigurationen für die Wahl, die verschlüsselten Stimmzettel der einzelnen Wähler, alle Zwischenergebnisse bei der Auswertung und das entschlüsselte Endergebnis. Die Trustees und der Authentication Server können auf das Bulletin Board schreiben und alle, auch externe Personen, können vom Bulletin Board lesen, aber vom Bulletin Board kann nichts gelöscht werden [31]. Dadurch hat das Bulletin Board unter anderem die Funktion, die Wahl zu dokumentieren.

Der *Authentication Server* steht zwischen den Wählern und dem Bulletin Board, da die Wähler nicht selbst auf das Bulletin Board schreiben können. Er überprüft die Stimmen der Wähler, bevor er sie auf das Bulletin Board schreibt [31].

Die *Election Authority* ist für alle administrativen Prozesse der Wahl zuständig. Durch sie wird die Konfiguration der Wahl definiert und auf das Bulletin Board geschrieben [31]. Außerdem ist die Election Authority für die Schlüsselgenerierung zuständig [31].

Die *Trustees* werten gemeinsam durch Multi-Party-Computation eine Wahl aus, wobei jeder Trustee einen Teil des privaten Schlüssels für die Entschlüsselung des Wahlergebnisses besitzt. Dabei ist jeder Trustee eine unabhängige Instanz und alle Trustees führen die gleichen Berechnungen zur Evaluierung einer Wahl aus. Da nicht garantiert ist, dass alle Trustees ehrlich sind, kontrollieren sie ihre Berechnungen gegenseitig durch nicht-interaktive Zero-Knowledge-Beweise [31].

2.2 Ablauf einer Wahl mit Ordinos

Dieser Abschnitt beschreibt die vier Phasen aus dem Paper [31], die Ordinos bei einer Wahl durchläuft und zeigt das Verhalten der oben eingeführten Komponenten.

Setup-Phase

In der Setup-Phase wird die Vorbereitung einer Wahl durchgeführt.

Für die kryptographischen Protokolle werden alle öffentlichen und privaten Schlüssel vom Authentication Server und den Trustees generiert und die öffentlichen Schlüssel auf das Bulletin Bord geschrieben. Außerdem wird festgelegt, welche Auswertungsfunktion mit welchen Konfigurationen verwendet wird und diese Informationen werden auf dem Bulletin Board gespeichert. Daraus leitet sich auch eine Vorlage für die Stimmabgabe der einzelnen Wähler ab, das ebenfalls auf dem Bulletin Board veröffentlicht wird. Im Allgemeinen ist auf dem Bulletin Board eine Liste an Kandidaten vorgegeben und der Wähler gibt seine Stimme ab, indem er den Kandidaten, die er wählt, eine verschlüsselte Eins und zu allen anderen Kandidaten eine verschlüsselte Null zuordnet. Je nach Art der Wahl kann die Stimmabgabe noch weitere Parameter oder Einschränkungen haben.

Abhängig von der spezifischen Wahl können hier noch weitere Vorberechnungen durchgeführt werden. Alles, was in der Setup-Phase festgelegt wird, kann im weiteren Verlauf der Wahl nicht mehr geändert werden.

Voting-Phase

In der Voting-Phase geben alle Wähler ihre Stimmen ab. Dabei schicken sie die verschlüsselten Stimmzettel an den Authentication Server, der überprüft, dass der Wähler wahlberechtigt ist und der die Stimmen entfernt, die leicht erkennbar ungültig sind. Wenn dies erfüllt ist, schreibt der Authentication Server die Stimme des Wählers auf das Bulletin Board.

Tallying-Phase

Die Tallying-Phase beginnt damit, dass alle Stimmen auf dem Bulletin Board mit Hilfe des homomorphen Paillier-Verschlüsselungsschemas aggregiert werden. Zuvor werden die Stimmen durch nicht-interaktive Zero-Knowledge-Beweise, die die Wähler mit den Stimmen abgeben, auf ihre Gültigkeit überprüft.

Anschließend werten die Trustees die Wahl mithilfe eines Evaluierungsprotokolls aus. Ein Evaluierungsprotokoll hat als Input die aggregierten Stimmen und gibt das entschlüsselte Ergebnis einer Wahl aus. Für Ordinos erforderlich ist, dass das Protokoll die Sicherheitseigenschaften Tally Hiding, Vote Privacy und Accountability erfüllt. Diese Begriffe werden in Abschnitt 2.3 erläutert. Jeder Trustee führt das gleiche Protokoll aus und die Trustees schreiben alle Zwischenergebnisse bei der Berechnung auf das Bulletin Board.

Ein Großteil dieser Arbeit bezieht sich auf die Tallying-Phase, da für diese Bachelorarbeit neue Evaluierungsprotokolle definiert werden, die von den Trustees ausgeführt werden können.

Verification Phase

In dieser Phase kann das Ergebnis der Wahl überprüft werden. Einerseits kann ein Wähler überprüfen, dass seine Stimme auf dem Bulletin Board liegt und korrekt in der Auswertung gezählt wurde. Andererseits kann jeder, insbesondere auch unabhängige Wahlbeobachter, kontrollieren, dass die Auswertung korrekt durchgeführt wurde, indem alle nicht-interaktiven Zero-Knowledge-Beweise und die Berechnung der Trustees kontrolliert werden.

2.3 Sicherheitseigenschaften von Ordinos

In diesem Kapitel werden die drei Sicherheitseigenschaften Tally Hiding, Vote Privacy und Accountability definiert und gezeigt, unter welchen Voraussetzungen diese in Ordinos erfüllt sind. Zuvor wird der k-risk-avoiding Angreifer beschrieben, der als Grundlage für die Definition der Sicherheit verwendet wird.

2.3.1 k-risk-avoiding Angreifer

Der stärkste Angreifer, gegen den Ordinos beweisbar sicher ist, ist der sogenannte k-risk-avoiding Angreifer [31]. Die Motivation für diese Entscheidung ist, dass ein typischer Angreifer neben dem eigentlichen Angriff auch das Ziel hat, möglichst unentdeckt zu bleiben. Ein k-risk-avoiding Angreifer darf maximal k Operationen durchführen [31]. Dabei könnte eine Operation beispielsweise sein, dass eine neue Stimme hinzugefügt wird oder eine Stimme entfernt wird. Jede dieser Operationen kann allerdings in der Verification Phase bemerkt werden, wodurch der Angreifer entdeckt werden kann. Dadurch ist der k-risk-avoiding Angriff ein Kompromiss zwischen einem erfolgreichen Angriff und einem unentdeckten Angriff.

Die k Operationen, die der Angreifer durchführen kann, werden außerhalb dieser Definition durch die unehrlichen Wähler simuliert, die der Angreifer kontrolliert [31]. Damit der Angriff mit großer Wahrscheinlichkeit unentdeckt bleibt, folgt daraus, dass es im Allgemeinen wenige unehrliche Wähler gibt.

2.3.2 Tally Hiding

Die Auflistung aller einzelnen Stimmen beziehungsweise aller aggregierter Stimmen sowie aller Zwischenergebnisse bei der Auswertung einer Wahl wird als (Full) Tally bezeichnet [31]. Die Eigenschaft Tally Hiding ist erfüllt, wenn die Tally nur in verschlüsselter Form vorliegt und nicht veröffentlicht wird. Stattdessen wird nur das Ergebnis entschlüsselt, das die minimal benötigten Informationen darstellt [31]. Am Beispiel einer Parlamentswahl ist das Ergebnis, welche Kandidaten einen Sitz bekommen. Andere Informationen wie den Stimmanteil pro Partei oder ob ein Kandidat einen Sitz über ein Direktmandat oder einen Listenplatz erhalten hat, bleiben geheim.

Das Tally Hiding hat den Vorteil, dass die Vote Privacy bei einer Wahl deutlich erhöht wird [31]. Insbesondere bei Wahlen mit wenigen Wählern ist es durch Veröffentlichung der Full Tally oft möglich, Rückschlüsse zu ziehen, welcher Wähler welchen Kandidaten gewählt hat. Doch auch bei großen politischen Wahlen bietet das Tally Hiding neben der verbesserten Vote Privacy weitere Vorteile: Wenn in einem Wahlkreis aus verschiedenen Kandidaten ein Abgeordneter gewählt wird, wie es beispielsweise bei der Wahl des britischen House of Commons in Abschnitt 4.1 beschrieben ist, wird beim Tally Hiding nur der finale Abgeordnete veröffentlicht, nicht die Stimmen pro Kandidat. Dadurch wird der gewählte Abgeordnete einerseits auch dann als legitimer Vertreter angesehen, wenn er nur knapp gewonnen hat und andererseits kommen Kandidaten, die unerwartet wenig Stimmen bekommen haben, nicht in unangenehme Situationen. Zusätzlich ist beispielsweise in einem Zweiparteiensystem nicht bekannt, wie knapp oder eindeutig das Ergebnis ist, sodass möglicherweise politischer Verödung in Wahlkreisen, die traditionell einen klaren Sieg für eine Partei haben, entgegengewirkt wird. Außerdem kann Tally Hiding dem Manipulieren von Wahlbezirksgrenzen gegensteuern und einige Angriffe wie Italian Attacks können verhindert werden [25].

2.3.3 Vote Privacy

Eine Voraussetzung, damit ein E-Voting System sinnvoll eingesetzt werden kann, ist die Vote Privacy. Intuitiv bedeutet es, dass nicht nachvollzogen werden kann, welche Partei ein Wähler gewählt hat. Dies wird dadurch formalisiert, dass es neben den in Abschnitt 2.1 beschriebenen Komponenten einen Angreifer gibt, der einen Wähler unter Beobachtung auswählt. Das Security Game [31] läuft folgendermaßen ab: Der Angreifer kontrolliert die unehrlichen Wähler und einen Teil der Trustees und der Wähler unter Beobachtung ist ehrlich und enthält sich nicht. Anschließend läuft die Wahl inklusive Auswertung durch, aber die unehrlichen Wähler und die Trustees unter Kontrolle des Angreifers dürfen sich so verhalten, dass möglichst das Wahlverhalten des Wählers unter Beobachtung herausgefunden wird. Nach der Veröffentlichung des Wahlergebnisses gibt der Angreifer den Kandidaten an, den der Wähler unter Beobachtung gewählt hat. Wenn der Angreifer den richtigen Kandidaten gefunden hat, gewinnt er das Security Game. Damit das E-Voting System sicher ist, darf jeder k-risk-avoiding Angreifer nur einen ”kleinen” Vorteil haben [31].

Intuitiv ist die Vote Privacy hoch, wenn der Wähler unter Beobachtung sich in der Masse der ehrlichen Wähler verstecken kann. Daraus folgt, dass die Vote Privacy besser ist, wenn mehr ehrliche Wähler an der Wahl teilnehmen und abhängig von der Evaluierungsfunktion ist.

Im Folgenden wird ein Theorem aus dem Paper [31] angegeben, das beschreibt, unter welchen Voraussetzungen Vote Privacy und Tally Hiding erfüllt sind.

Theorem 1 (Tally Hiding und Vote Privacy)
Tally Hiding und Vote Privacy sind bezüglich eines k-risk-avoiding Angreifers erfüllt, wenn die folgenden Eigenschaften gelten [31]:

(a) Das verwendete Verschlüsselungsschema und die Signaturen sind sicher nach den jeweiligen Security Games.

(b) Die Protokolle für die Schlüsselgenerierung und die Verschlüsselung sind nicht-interaktive Zero-Knowledge-Beweise.

(c) Eine ausreichende Anzahl an ehrlichen Wählern enthält sich nicht. Wie viele Wähler ausreichend sind, ist stark von der Evaluierungsfunktion abhängig und wird in dieser Arbeit nicht weiter betrachtet.

(d) Der Angreifer darf nur die unehrlichen Wähler und weniger Trustees als für die Entschlüsselung notwendig sind kontrollieren.

(e) Die Evaluierungsprotokolle bekommen als Input die Chiffretexte und veröffentlichen bei der Evaluierung nur das Ergebnis, jedoch keine weiteren Informationen.

2.3.4 Accountability

Ein Nachteil des Tally Hiding ist, dass nicht mehr einfach überprüft werden kann, ob das von den Trustees berechnete Ergebnis korrekt ist, da nur das Ergebnis entschlüsselt vorliegt. Daher kann Tally Hiding in einem E-Voting System nicht alleine eingesetzt werden, sondern es muss zusätzlich die Eigenschaft Verifiability erfüllt sein. Verifiability bedeutet, dass jeder Wähler verifizieren kann, dass seine Stimme korrekt in der Auswertung gezählt wurde und dass jeder überprüfen kann, dass das Ergebnis aus den vorliegenden Stimmen korrekt berechnet wurde [31]. Dadurch kann geschlossen werden, dass das Ergebnis insgesamt korrekt ist. Ordinos erfüllt nicht nur Verifiability, sondern auch die stärkere Eigenschaft Accountability. Diese besagt, dass wenn eine Manipulation bei der Berechnung des Ergebnisses stattfindet, nicht nur die Manipulation entdeckt werden kann, sondern auch erkannt werden kann, wer die Manipulation verursacht hat, sodass dieser zur Rechenschaft gezogen werden kann [25].

In Paper [31] geben Küsters et al. ebenfalls ein Theorem für Accountability und Verifiability an.

Theorem 2 (Accountability)
Accountability und dadurch auch Verifiability sind in Ordinos erfüllt, wenn neben einigen der in Theorem 1 genannten Voraussetzungen auch das Folgende gilt:

(a) Das Bulletin Board, die Verifikationsgeräte der Wähler sowie einige für den Beweis notwendigen theoretischen Instanzen sind ehrlich.

(b) Bei den von den Trustees verwendeten Multi-Party-Computation-Protokollen kann öffentlich identifiziert werden, wenn der Output nicht das korrekte Ergebnis des Inputs ist.

2.4 Grundbausteine von Ordinos

In Ordinos wird das homomorphe Paillier-Verschlüsselungsschema verwendet [31]. Dieses definiert bereits die Algorithmen für die Ver- und Entschlüsselung. Bei der Entschlüsselung wird die verteilte Variante des Paillier-Verschlüsselungsschemas verwendet, bei der jeder Trustee einen Teil des privaten Schlüssels besitzt und mehrere Trustees benötigt werden, um einen Chiffretext zu entschlüsseln. Durch das Paillier-Verschlüsselungsschema können Klartexte verschlüsselt addiert werden, doch für komplexe Evaluierungsprotokolle sind weitere arithmetische und logische Grundoperationen nötig. Diese lassen sich ebenfalls als Multi-Party-Computation-Protokolle mit der Paillier-Verschlüsselung realisieren und sind im Folgenden beschrieben. Alle Protokolle erhalten dabei als Input verschlüsselte Zahlen und geben das Ergebnis ebenfalls als Chiffretext aus. Für all diese Protokolle ist bereits bewiesen, dass sie korrekt sind und alle für Ordinos relevanten Sicherheitseigenschaften erfüllen [25].

2.4.1 Arithmetische-logische Grundbausteine

In Ordinos wird die *Addition* zweier Chiffretexte verwendet, die schon im homomorphen Paillier-Verschlüsselungsschema enthalten ist. Da eine Addition der Klartexte einer Multiplikation der Chiffretexte entspricht, ist die Addition sehr schnell. Im Pseudocode wird eine Addition folgendermaßen dargestellt: $\mathrm{E}_{\mathrm{pk}}(c) = \mathrm{E}_{\mathrm{pk}}(a) +_e \mathrm{E}_{\mathrm{pk}}(b)$, wobei $c = a + b$ und $\mathrm{E}_{\mathrm{pk}}(x)$ einen Chiffretext bezeichnet, der eine Verschlüsselung des Klartextes x ist.

Auch für die *Multiplikation* existiert in Ordinos bereits ein Multi-Party-Computation-Protokoll [42]. Das Protokoll setzt die Multiplikation zweier verschlüsselter Werte durch die mehrfache Multiplikation eines verschlüsselten Wertes mit einem bekannten Wert um, da letzteres durch Additionen erreicht werden kann. Die Multiplikation wird im Pseudocode als $\mathrm{E}_{\mathrm{pk}}(c) = \mathrm{E}_{\mathrm{pk}}(a) \cdot_e \mathrm{E}_{\mathrm{pk}}(b)$ dargestellt, wobei $c = a \cdot b$ gilt. Um zwei verschlüsselte Werte $\mathrm{E}_{\mathrm{pk}}(a)$ und $\mathrm{E}_{\mathrm{pk}}(b)$ zu multiplizieren wird folgendermaßen vorgegangen [42]: Jeder Teilnehmer der Berechnung wählt zuerst eine zufällige Zahl d_i, verschlüsselt diese und schreibt den verschlüsselten Wert $\mathrm{E}_{\mathrm{pk}}(d_i)$ auf das Bulletin Board. Anschließend werden alle Zufallswerte und der ersten Faktor der Multiplikation verschlüsselt addiert und das Ergebnis $\mathrm{E}_{\mathrm{pk}}(m) = \mathrm{E}_{\mathrm{pk}}(a + d_1 + \cdots + d_n)$ wird entschlüsselt. Die Entschlüsselung gibt keine Informationen über den ursprünglichen Faktor a preis, da auf a Zufallswerte addiert wurden und kein Teilnehmer der Berechnung alle d_is kennt. Daher sieht der entschlüsselte Wert zufällig aus, unabhängig davon, welches a vorliegt. Man nennt diesen Vorgang Blinding, da der Klartext maskiert wurde. Zusätzlich berechnet jeder Teilnehmer den Wert $\mathrm{E}_{\mathrm{pk}}(d_i \cdot b)$. Dies ist durch Addition möglich, da jeder Teilnehmer sein d_i als Klartext kennt.

Aus diesen Werten kann durch m Additionen der Wert $\mathrm{E}_{\mathrm{pk}}(a \cdot b + d_1 \cdot b + \cdots + d_n \cdot b)$ berechnet werden. Davon können nun die Reste $d_i \cdot b$ durch verschlüsselte Subtraktion abgezogen werden und man erhält das gewünschte Ergebnis $\mathrm{E}_{\mathrm{pk}}(a \cdot b)$. Das Multi-Party-Computation-Protokoll für die Multiplikation dauert länger als eine Addition, da mehrere Additionen und Subtraktionen durchgeführt werden müssen und die Teilnehmer der Berechnung einige der Werte untereinander austauschen müssen.

Zusätzlich stellt Ordinos ein Protokoll für den Test auf *Gleichheit* zur Verfügung. Das Ergebnis des Tests ist $\mathrm{E}_{\mathrm{pk}}(1)$, wenn $a = b$ ist und ansonsten $\mathrm{E}_{\mathrm{pk}}(0)$ und wird im Pseudocode als $\mathrm{E}_{\mathrm{pk}}(bool) = \mathrm{eq}(\mathrm{E}_{\mathrm{pk}}(a), \mathrm{E}_{\mathrm{pk}}(b))$ dargestellt. Da das Paillier-Verschlüsselungsschema nur ganze

Zahlen als Klartext darstellen kann, ist der Datentyp Boolean in Ordinos nicht vorhanden. Wenn Wahrheitswerte benötigt werden, werden daher die Zahlen ”null” bzw.”eins” verschlüsselt. Das Protokoll für den Vergleich folgt dem Paper von Lipmaa et al. und nachfolgend wird die Grundidee für den Vergleich einer Zahl x mit Null beschrieben. [32]: Zuerst wird ein verschlüsselter Zufallswert r benötigt, dessen Bit-Darstellung ebenfalls verschlüsselt vorliegt und der invertierbar in $\mathbb{Z}_n$ ist. Dieser kann bereits in der Setup-Phase berechnet werden.

Für den eigentlichen Vergleich wird x maskiert, indem ein Wert m als Summe von r und x verschlüsselt berechnet wird und der Wert m entschlüsselt wird. Intuitiv gesehen ist $x = 0$, wenn $r = m$ ist. Dieser Test wird nun verschlüsselt durchgeführt: Von r ist die Bit-Darstellung bekannt und von m kann sie berechnet werden, da m als Klartext vorliegt. Aus den Bit-Darstellungen der beiden Werte kann nun verschlüsselt die Hamming-Distanz berechnet werden. Diese gibt für zwei Zahlen an, wie viele Bits unterschiedlich sind. Nun steht verschlüsselt fest, ob $r = m$ gilt. Da r invertierbar ist, lässt sich dieses Ergebnis auf x zurückrechnen.

Das Protokoll für den Vergleich ist deutlich aufwändiger als die zuvor vorgestellten Protokolle. Dies liegt vor allem daran, dass die Berechnungen bitweise durchgeführt werden müssen, und dass für die Berechnung der Hamming-Distanz ein Polynom benötigt wird, dessen Anzahl an Koeffizienten der Bitlänge entspricht. Zusätzlich sind durch das Maskieren weitere Operationen nötig.

Außerdem hat Ordinos einen *größer-gleich-Vergleich* zwischen zwei Chiffretexten $\mathrm{E}_{\mathrm{pk}}(a)$ und $\mathrm{E}_{\mathrm{pk}}(b)$. Das Ergebnis dieses Vergleiches ist $\mathrm{E}_{\mathrm{pk}}(1)$, wenn $a \geq b$ ist und $\mathrm{E}_{\mathrm{pk}}(0)$, wenn $a < b$ ist. Im Pseudocode wird dies wie folgt geschrieben: $\mathrm{E}_{\mathrm{pk}}(bool) = \mathrm{gt}(\mathrm{E}_{\mathrm{pk}}(a), \mathrm{E}_{\mathrm{pk}}(b))$. Ein größer-gleich-Vergleich wird durchgeführt, indem rekursiv etwa $\log(n)$ Vergleiche aufgerufen werden. Somit ist ein größer-gleich Vergleich abhängig von der maximalen Bitlänge der Klartexte der zu vergleichenden Zahlen und benötigt länger als ein gleich-Vergleich mit ähnlicher Bitlänge.

Auch dieses Protokoll ist aus Paper [32] und die Grundidee wird in diesem Absatz erklärt. Anstatt direkt a und b miteinander zu vergleichen, wird die um die maximale Bitlänge verschobene Differenz z der beiden Zahlen miteinander verglichen. Anschließend wird diese Differenz mit einer Zufallszahl r maskiert und entschlüsselt, um die maskierte Differenz m in eine obere und eine untere Hälfte aufgeteilt, wobei jede Hälfte gleich viele Bits hat. Nun wird auf den oberen Hälften von r und m ein Gleichheits-Test durchgeführt. Wenn eine der beiden Hälften größer ist, ist bekannt, welche der Zahlen insgesamt größer ist. Nur wenn beide gleich sind, wird der gesamte Vorgang rekursiv mit den unteren Hälften durchgeführt.

Die größer-gleich-Vergleiche benötigen von den vorgestellten Operationen am meisten Laufzeit. Dies liegt einerseits an der Rekursion und andererseits daran, dass auf jeder Rekursionsstufe ebenfalls ein Gleichheits-Test durchgeführt werden muss.

Für die Einschätzung der Laufzeit in Ordinos sind somit vor allem die Anzahl der Vergleiche ausschlaggebend. Wenn Vergleiche dennoch nötig sind, sollten wenn möglich gleich-Vergleiche verwendet werden und die zugehörigen Klartexte sollten möglichst klein sein. Ziel ist es, so viele Teilschritte wie möglich durch Additionen und Multiplikationen umzusetzen. Multiplikationen sind zwar teurer als Additionen, tragen aber noch immer deutlich weniger zur Laufzeit bei als Vergleiche.

Jede der Evaluierungsfunktionen ist eine Hintereinanderausführung der vorgestellten Multi-Party-Computation-Protokolle. Dadurch ist es theoretisch möglich, beliebige Algorithmen umzusetzen, aber die Herausforderung liegt darin, diese Algorithmen effizient zu gestalten.

Algorithmus 2.1 ifThenElse

procedure IFTHENELSE($E_{pk}(condition)$,$E_{pk}(value1)$, $E_{pk}(value1)$)
  return $E_{pk}(condition) \cdot_e E_{pk}(value1) +_e$ NEGATE($E_{pk}(condition)$)$\cdot_e E_{pk}(value2)$
end procedure

2.4.2 Kontrollstrukturen

Da durch das Tally Hiding alle Zwischenergebnisse verschlüsselt bleiben müssen, sind Kontrollstrukturen bei der Programmierung mit Ordinos anders zu behandeln als in der klassischen Programmierung. Die Kontrollstrukturen benötigen im Allgemeinen mehr Laufzeit als bei Programmen, die kein Tally Hiding nutzen, da die Abbruchbedingungen im Allgemeinen nur in verschlüsselter Form vorliegen. Dies wird am Beispiel der for-Schleife, der while-Schleife und der if-Abzweigung gezeigt und erklärt, welche Einflüsse sie auf die Programmierung haben.

Schleifen

Bei for-Schleifen wird schrittweise bis zu einer Obergrenze iteriert. Beim Tally Hiding kann diese Obergrenze eine bekannte, nicht verschlüsselte Zahl oder ein verschlüsselter Wert sein. Ist die Obergrenze ein bekannter Wert, wie beispielsweise die Anzahl an Wahlkreisen, kann die Schleife normal angewandt werden. Ist die Obergrenze jedoch verschlüsselt, kann der Trustee nicht wissen, wie viele Schleifeniterationen er durchführen muss. Daher muss eine nicht verschlüsselte obere Schranke für die Anzahl an Iterationen festgelegt werden, sodass alle benötigten Schleifeniterationen durchgeführt werden, und das korrekte Ergebnis gespeichert wird.

Bei while-Schleifen wird analog vorgegangen, indem eine obere Schranke für die Anzahl an Iterationen gefunden werden muss und das Ergebnis, solange die Bedingung erfüllt ist, gespeichert wird.

In beiden Fällen ist es wichtig, dass die obere Schranke möglichst klein ist, um unnötige Iterationen zu vermeiden. Dies ist vor allem bei realen politischen Wahlen entscheidend, um akzeptable Laufzeiten zu realisieren, denn bei solchen Wahlen sind alle bekannten Größen wie die Anzahl an Wahlkreisen, die Sitzzahl oder die Anzahl an gültigen Stimmen eher groß, weshalb auch im Idealfall im Allgemeinen viele Iterationen notwendig sind.

Bedingungen

Solange bei if-Verzweigungen nicht entschlüsselt wird, ob eine Bedingung erfüllt ist, oder nicht, muss von beiden Fällen ausgegangen werden und für beide Fälle das Ergebnis berechnet werden. Anschließend wird durch Additionen und Multiplikationen das gewünschte Ergebnis extrahiert. Dies ist in Pseudocode 2.1 zu sehen.

Auswirkungen auf das Programmieren

Bei beiden Kontrollstrukturen zeigt sich, dass jeder Pfad, den das Programm nehmen könnte, ausgeführt werden muss, denn bei verschlüsselten Bedingungen ist unbekannt, welcher Pfad verwendet wird. In Bezug auf die Effizienz von Programmen folgt daraus, dass es notwendig ist, jeden Pfad für die Optimierung zu betrachten. In der klassischen Programmierung wird bei Verzweigungen meist der Fokus auf die Laufzeit-Optimierung der wahrscheinlicheren Fälle gelegt. Wenn es einen Fall gibt, der nur mit sehr kleiner Wahrscheinlichkeit eintritt, ist es unproblematisch, wenn dieser unverhältnismäßig viel Laufzeit benötigt. Bei der Programmierung mit Tally Hiding hingegen müsste dieser unwahrscheinliche Fall in jeder Ausführung des Programms ebenfalls durchlaufen werden. Daher ist es wichtig, dass auch dieser nicht übermäßig viel Laufzeit beansprucht. Daraus folgt, dass die Laufzeit von bekannten Algorithmen ohne weitere Anpassungen mindestens der Laufzeit des Worst Case entspricht.

3 Verwandte Arbeiten

In der Forschung wurden bereits mehrere E-Voting Systeme entwickelt, die kryptographische Eigenschaften wie Verifiability sicherstellen. Das bekannteste Beispiel hierfür ist das Web-basierte E-Voting System Helios [5], auf dessen Ideen auch Ordinos aufgebaut ist [25]. Doch die Eigenschaft, die Ordinos besonders auszeichnet, ist das Tally Hiding. Auch für das Tally Hiding wurde bereits Ergebnisse in zwei Richtungen veröffentlicht. Einerseits wurde an Systemen gearbeitet, die den Teil der Tally verstecken, der für bestimmte Angriffe relevant ist, um diesen Angriffen entgegenzuwirken [28]. Andererseits gibt es E-Voting Systeme wie Ordinos, die die Full Tally verstecken. Dafür wurden zuerst Grundlagen erarbeitet, um benötigte Operationen wie Multiplikationen, Vergleiche und eine verteilte Entschlüsselung zu realisieren [33], [12] und dabei Eigenschaften wie Verifiability und Tally Hiding zu erhalten. Mit Hilfe dieser Grundlagen wurden sichere Auswertungen im Bereich der E-Voting Systeme ermöglicht. Darauf aufbauend wurden Evaluierungsfunktionen entwickelt, die unterschiedliche Arten von Auswertungen wie Mehrheitswahlen, Borda oder Condorcet ermöglichen [10], [25].

In dieser Phase befindet sich auch Ordinos. Die theoretischen Grundlagen für Ordinos wurden im Paper von Küsters et al. gelegt [31]. Dabei wurden die einzelnen Komponenten des E-Voting Systems entwickelt, doch der Fokus lag auf den Eigenschaften von Ordinos. Es wurde gezeigt unter welchen Voraussetzungen Tally Hiding, Vote Privacy und Accountability erfüllt sind und dass diese Voraussetzungen bei Ordinos gegeben sind. Außerdem wurde untersucht, welche Auswirkungen Tally Hiding auf die Sicherheit des Systems hat. Gleichzeitig wurden erste Evaluierungsfunktionen implementiert. Diese waren recht simpel und haben beispielsweise ein Ranking der Kandidaten oder einen Gewinner zurückgegeben [31]. In der Arbeit von Hertel et al. [25] wurde Ordinos um mehrere Wahlverfahren erweitert, indem Borda, Hare-Niemeyer, Condorcet und Instant-Runoff-Voting hinzugefügt wurden. Diese Wahlen sind deutlich komplizierter als nur die Berechnung eines Gewinners und bieten teils mehrere Optionen der Auswertung an. So wurde gezeigt, dass mit Ordinos auch kompliziertere Wahlen möglich sind, und diese in relativ kurzer Zeit ausgewertet werden können. Mit der Aggregation von gerankten Stimmzetteln beim Instant-Runoff-Voting wurde auch eine Grenze von Ordinos entdeckt, denn es hat sich gezeigt, dass eine effiziente Aggregation solcher Stimmzettel nur bis etwa fünf Kandidaten möglich ist [25]. Diese Bachelorarbeit knüpft an den aktuellen Stand von Ordinos an und untersucht, inwiefern sich Ordinos für politische Wahlen aus der Praxis eignet. Dabei wird nicht mehr von Details abstrahiert, sondern die Wahlen werden genau so mit Ordinos umgesetzt, wie sie in der Praxis durchgeführt werden.

4 Beschreibung einiger politischer Wahlverfahren

In diesem Kapitel werden verschiedene politische Wahlverfahren genau beschrieben und die Verfahren ausgewählt, die in Kapitel 6 mit Ordinos umgesetzt werden. Motivation der Arbeit ist es, Ordinos um Wahlverfahren zu ergänzen, die in der Praxis angewandt werden, insbesondere um zu zeigen, wie mit den Feinheiten einer realen Wahl sowie der Vermischung von Wahlsystemen umgegangen werden kann. Für diesen Zweck eignen sich große politische Wahlen sehr gut, da diese beispielsweise durch Wahlkreise, Minderheitsparteien und Sorge vor Zersplitterung eines Parlaments kompliziertere Ausgangsbedingungen haben. Dadurch ist teils erforderlich, generische Wahlverfahren anzupassen, um sie auf die Bedürfnisse zuzuschneiden. Bei kleineren Wahlen, wie beispielsweise Vereinswahlen, ist dieser Aufwand unverhältnismäßig hoch.

Es gibt zwar auch nicht-politische Organisationen, die kompliziertere Wahlverfahren verwenden, doch diese haben meist ebenfalls keine Änderungen oder Ergänzungen zu dem theoretischen Vorgehen der jeweiligen Wahl. Beispielsweise das Debian-Projekt und die Piratenpartei nutzen eine der Condorcet-Varianten [13], [37], doch hier gibt es keine an die Organisation angepasste Änderung der Grundmethode und viele Condorcet-Varianten sind bereits in Ordinos implementiert [24]. An der Universität Stuttgart werden die Sitze bei Verhältniswahlen nach dem d'Hondt-Verfahren verteilt und bei Gleichstand wird durch Los entschieden [41], doch auch hier wurden keine Änderungen zum theoretischen Verfahren gemacht und das d'Hondt-Verfahren ist sehr ähnlich zum Sainte-Laguë-Verfahren, das im Zuge dieser Arbeit mit Ordinos umgesetzt wird. Ein weiteres Beispiel, in dem zwar Mehrheitswahlen abgeändert werden, um sie spezifisch auf die vorliegenden Bedingungen zuzuschneiden, ist Jury-Voting. Da dabei oft eine kleine Gruppe an Personen, beispielsweise Geschworene, über eine Sache von großer Bedeutung abstimmen müssen, gibt es hier strengere Regeln wie beispielsweise Einstimmigkeit oder eine Zweidrittel-Mehrheit. Ein Beispiel hierfür sind die Wahlen der Richter des Bundesverfassungsgerichts [23]. Doch trotz der Änderungen haben diese Wahlen den Nachteil, dass sie relativ einfach sind und daher nicht den Anforderungen für diese Bachelorarbeit entsprechen.

Bei der Auswahl der politischen Wahlen wird darauf geachtet, dass möglichst unterschiedliche Wahlen enthalten sind. Dies bezieht sich einerseits auf den grundlegenden Wahltyp wie relative Mehrheitswahl, absolute Mehrheitswahl, Verhältniswahl und Single Transferable Vote. Andererseits wird auch darauf geachtet, Mischwahltypen wie die personalisierte Verhältniswahl und unterschiedliche Ergänzungen des gleichen Wahltyps einzubeziehen.

4.1 House of Commons (Vereinigtes Königreich)

Das House of Commons hat 650 Abgeordnete, die durch eine relative Mehrheitswahl gewählt werden [18]. Jeder Wähler hat eine Stimme für einen Kandidaten im Wahlkreis und der Kandidat, der die meisten Wählerstimmen in einem Wahlkreis erhält, wird Abgeordneter und vertritt den Wahlkreis im Parlament [18]. Daher gibt es genau 650 Wahlkreise. Da die Aufstellung eines Kandidaten mit einer Zahlung verbunden ist, gibt es verhältnismäßig wenig Kandidaten pro Wahlkreis [21].

Im Falle eines Gleichstandes zwischen mehreren Kandidaten in einem Wahlkreis entscheidet der Wahlleiter des jeweiligen Wahlkreises, wer die Wahl gewinnt [29]. Der Wahlleiter ist hauptverantwortlich für die Durchführung einer Wahl und muss das Wahlergebnis verkünden [40]. Es ist nicht vorgegeben, wie der Wahlleiter einen Gleichstand auflöst und bei Wahlen des House of Commons kam bislang kein Gleichstand vor [29]. Allerdings gilt dieselbe Regelung im Vereinigten Königreich bei lokalen Wahlen und dort wurde das Tie-Breaking immer durch einen Zufallsmechanismus wie das Werfen einer Münze gelöst [29]. Daher ist auch bei einem Gleichstand für die Parlamentswahl Tie-Breaking nach Zufall zu erwarten.

Die Wahl das House of Commons folgt einem einfachen Prinzip. Sie wird trotzdem in Ordinos implementiert, um zu zeigen, dass sich auch unkomplizierte Wahlen mit Ordinos umsetzen lassen.

4.2 Präsident der USA

Die Wahl des US-Präsidenten ist eine Mehrheitswahl und erfolgt indirekt über Wahlpersonen [19]. Zuerst werden in jedem der 50 Bundesstaaten und dem District of Columbia die Wahlpersonen gewählt, wobei die Anzahl an Wahlpersonen pro Bundesstaat bereits vor der Wahl feststeht. Dies geschieht meist nach dem Prinzip ”winner-takes-all”, das heißt die Partei mit den meisten Stimmen schickt alle Wahlpersonen zum Wahlpersonengremium [19]. Ausnahmen hiervon sind die Bundesstaaten Maine und Nebraska, bei denen das ”winner-takes-all” Prinzip durch Instant-Runoff-Voting beziehungsweise eine relative Mehrheitswahl ergänzt wird [19]. Wenn es bei der Wahl der Wahlpersonen einen Gleichstand gibt, sind die Regeln für das Tie-Breaking von Bundesstaat zu Bundesstaat unterschiedlich. Meist wird durch Zufall entschieden oder die Wahl wiederholt [39].

Anschließend versammeln sich die 538 Wahlpersonen im Wahlpersonengremium und stimmen durch Mehrheitswahl ab, wer Präsident wird [19]. Dabei sind die Wahlpersonen nicht in allen Bundesstaaten an das Ergebnis des Bundesstaates gebunden, den sie vertreten. [19]. Der Kandidat wird Präsident, der die absolute Mehrheit der Stimmen der Wahlpersonen bekommt. Bei einem Gleichstand in diesem Schritt wird der Präsident stattdessen im Repräsentantenhaus gewählt [19]. Dort ist die Verteilung anders, da jeder Bundesstaat genau eine Stimme hat [19].

Die Wahl des US-Präsidenten wird im Zuge dieser Bachelorarbeit nicht in Ordinos umgesetzt. Sie ist verhältnismäßig einfach, da es sich hauptsächlich um eine Hintereinanderausführung schon implementierter Wahlverfahren handelt, und durch die Wahl des House of Commons wird bereits eine einfache Wahl implementiert, an der die Ergänzungen zu Wahlverfahren aus der Praxis gezeigt werden können. Aus den gleichen Gründen werden auch die Wahlen des französischen Präsidenten und der französischen Nationalversammlung in dieser Arbeit nicht in Ordinos umgesetzt.

4.3 Präsident von Frankreich

Der französische Präsident wird durch eine absolute Mehrheitswahl gewählt [17]. Dafür gibt es zwei Wahlgänge. Im ersten Wahlgang wird eine Mehrheitswahl zwischen allen Kandidaten durchgeführt und wenn ein Kandidat mehr als 50% der Stimmen erhält, hat er die Wahl gewonnen und ist Präsident. Wenn kein Kandidat eine absolute Mehrheit der Stimmen erhält, wird der zweite Wahlgang durchgeführt. Dieser ist eine Stichwahl zwischen den beiden Kandidaten mit den meisten Stimmen aus dem ersten Wahlgang. Der Kandidat, der hier eine einfache Mehrheit der Stimmen erhält, wird Präsident [17].

4.4 Nationalversammlung (Frankreich)

Die französische Nationalversammlung besteht aus 577 Abgeordneten, wobei jeder Abgeordnete genau einen Wahlkreis vertritt. Die Wahl der Abgeordneten findet strikt nach Wahlkreis getrennt statt und ist eine Mehrheitswahl [16]. In jedem Wahlkreis wird wie bei der Wahl des französischen Präsidenten in zwei Wahlgängen gewählt. Im ersten Wahlgang treten alle Kandidaten an. Ein Kandidat gewinnt im ersten Wahlgang direkt, wenn er die absolute Mehrheit der Stimmen hat und zusätzlich mindestens 25% der Wahlberechtigten für ihn gestimmt haben [20]. Wenn kein Kandidat im ersten Wahlgang gewonnen hat, wird ein zweiter Wahlgang durchgeführt. An diesem nehmen nur die Kandidaten teil, die im ersten Wahlgang den ersten oder zweiten Platz gewonnen haben oder mindestens $12,5\%$ der Stimmen erhalten haben [20]. Den zweiten Wahlgang gewinnt der Kandidat mit den meisten Stimmen.

4.5 Volksabstimmungen

Volksabstimmungen sind Wahlen, bei denen ein neuer Gesetzestext oder ähnliches vorliegt und die Bürger entscheiden, ob sie diesen befürworten oder ablehnen. Dabei gibt es oft zusätzliche Mechanismen neben einer einfachen Mehrheitswahl, die sicherstellen sollen, dass das gesamte Volk hinter der Entscheidung steht und nicht nur ein Teil des Volkes. Letzteres könnte beispielsweise eintreten, wenn die Wahlbeteiligung gering ist oder die Meinungen der Wähler ungleich über die Wahlbezirke verteilt sind.

In Baden-Württemberg benötigen Volksabstimmungen ein Quorum [44]. Das bedeutet, dass eine Entscheidung durch eine einfache Mehrheit getroffen wird. Allerdings ist die Entscheidung für ein Gesetzt nur dann gültig, wenn mindestens 20% der Wahlberechtigten für dieses Gesetz gestimmt haben [44].

In der Schweiz gibt es für wichtige Entscheidungen wie beispielsweise Verfassungsänderungen ein obligatorisches Referendum [45]. Das heißt, um die Verfassungsänderung durchzuführen, muss sowohl das Volks-mehr als auch das Stände-mehr gelten. Das Volks-mehr ist erfüllt, wenn die Mehrheit der Wähler der Änderung zustimmt und das Stände-mehr ist erfüllt, wenn die Mehrheit der Kantone zustimmt. Ein Kanton stimmt der Verfassungsänderung zu, wenn in diesem Kanton das Volks-mehr herrscht [45].

Volksabstimmungen werden in dieser Arbeit nicht umgesetzt, da sie zu ähnlich zu einfachen Mehrheitswahlen sind.

4.6 Europäisches Parlament

Das Europäische Parlament wird nach einer Verhältniswahl gewählt [47]. Dabei stehen jedem Mitgliedsstaat der Europäischen Union eine festgelegte Anzahl an Sitzen zur Verfügung, die abhängig von der Größe des Staates sind [47]. Jeder Mitgliedsstaat füllt die ihm zustehenden Sitze durch eine Verhältniswahl [47]. Für die Verhältniswahlen gibt es verschiedene Vorgaben: Basis für die Verhältniswahl müssen entweder Listen oder das Single-Transferable-Vote-System sein, wobei die Listenplätze vom Wähler verändert werden dürfen [47]. Es darf eine Sperrklausel geben, aber diese muss unter 5% bleiben und Wahlkreise sind erlaubt, unter der Bedingung, dass die Elemente der Verhältniswahl erhalten bleiben [47].

Trotz dieser Einschränkungen sind die Wahlverfahren in den Mitgliedsstaaten sehr unterschiedlich. Schon der Typ der Verhältniswahl unterschiedet sich zwischen den Staaten und es werden insgesamt die Verfahren d'Hondt, Hare-Niemeyer, Sainte-Laguë (klassisch und modifiziert), Enishimeni Analogiki, Single-Transferable-Vote, Hagenbach-Bischoff und Quotenverfahren mit Droop-Quote angewandt [47]. Weiter variieren die Sperrklausel und die Möglichkeit zu Panaschieren oder die Listenreihenfolge zu ändern und einige Staaten verwenden Wahlkreise, in denen teils ein anderes Verfahren für die Unterverteilung verwendet wird [47]. Dies beschreibt nur den groben Überblick, da die einzelnen Staaten noch weitere Sonderregelungen in ihren Wahlen hinzufügen können.

Daher wurde die Europawahl nicht in Ordinos umgesetzt, da dafür die Wahlen von 27 Staaten umgesetzt werden müssten, was den Rahmen dieser Arbeit sprengt.

4.7 Dáil Éireann (Irland)

Das irische Parlament wird durch eine Verhältniswahl mit Single Transferable Vote gewählt [36]. Die Wähler geben in ihrem Stimmzettel ein Ranking der Kandidaten ab, wobei nicht allen Kandidaten ein Rang zugeteilt werden muss. Mit dem Ranking drückt ein Wähler aus, an welchen Kandidaten seine Stimme weitergegeben wird, wenn sie nicht für den erstpräferenzierten Kandidaten verwendet werden kann [36].

Sowohl die Wahl als auch die Auswertung werden in Wahlkreise eingeteilt und jedem Wahlkreis stehen eine festgelegte Anzahl von Sitzen zur Verfügung [11]. Die Auswertung pro Wahlkreis findet in verschiedenen Runden statt, wobei in jeder Runde entweder Kandidaten gewählt oder ausgeschlossen werden [36]. Für die erste Runde wird die Anzahl an Erstpräferenzen pro Kandidat aufsummiert. Anschließend wird abhängig von der Sitzzahl und der Anzahl an gültigen Stimmen eine Quote berechnet. Jeder Kandidat, der diese Quote erreicht, ist als Abgeordneter ins Parlament gewählt. Die Anzahl an Stimmen, um die ein Kandidat die Quote übertrifft, werden als Surplus Votes bezeichnet und an die noch verbleibenden Kandidaten verteilt. Die Verteilung der Surplus Votes ist proportional dazu, wen die entsprechenden Wähler im nächst-unteren Rang gewählt haben. Wenn innerhalb einer Runde kein Kandidat die genug Stimmen hat, um die Quote zu erreichen,

wird der Kandidat gestrichen, der am wenigsten Stimmen hat. Die Stimmen für den gestrichenen Kandidaten werden ebenfalls auf die anderen Kandidaten verteilt [36]. Dieser Vorgang wird so lange wiederholt, bis alle Parlamentssitze vergeben sind.

Mit Instant-Runoff Voting wurde in Ordinos bereits ein ähnliches Wahlsystem implementiert, das ebenfalls Rankings als Stimmzettel nutzt [25]. Hier wurde festgestellt, dass das Verfahren nur mit bis zu fünf Kandidaten zu akzeptablen Laufzeiten führt und dass die hohen Laufzeiten hauptsächlich durch die Aggregation der Stimmzettel mit Rankings anfallen. Aus dem Paper [25] folgt, dass Stimmzettel mit Rankings eine Maximalgröße von fünf Kandidaten haben dürfen, damit die Aggregation mit akzeptablen Laufzeiten möglich ist. Dieses Problem trifft auch auf die Wahl des irischen Parlaments zu. Da die Anzahl an Kandidaten pro Wahlkreis in Irland zwischen 9 und 20 variiert [11], kann das Verfahren nicht mit akzeptabler Laufzeit in Ordinos implementiert werden.

4.8 Sainte-Laguë-Verfahren

Das Sainte-Laguë-Verfahren ist ein Sitzzuteilungsverfahren, das heißt eine gegebene Anzahl an Sitzen n_{seats} wird auf $n_{parties}$ Parteien in Abhängigkeit der Stimmen pro Partei verteilt. Da es für mehrere Wahlen wie die Bundestagswahl und die Wahl des norwegischen Parlaments relevant ist, wird das Verfahren im folgenden Abschnitt als einzelner Baustein beschrieben. Das Sainte-Laguë-Verfahren wird außerdem in Kapitel 6 in Ordinos umgesetzt. Dafür werden zusätzlich zur Beschreibung noch einige Eigenschaften des Verfahrens erklärt.

4.8.1 Berechnung einer Sitzverteilung mit Sainte-Laguë

Für die Berechnung einer Sitzverteilung nach Sainte-Laguë gibt es verschiedene Algorithmen, deren Ergebnis äquivalent ist [46]. Im Folgenden werden das Höchstzahlverfahren und das Divisorverfahren vorgestellt.

Beim Höchstzahlverfahren werden die aggregierten Stimmen pro Partei durch $2i - 1$ für $i = 1, 2, \ldots$ geteilt und die Quotienten, die auch als Höchstzahlen bezeichnet werden, der Größe nach absteigend sortiert. Die Anzahl an Höchstzahlen, die einer Partei innerhalb der n_{seats} größten Höchstzahlen zugeordnet sind, ergibt die Anzahl an Sitze, die dieser Partei zustehen [15]. Anders formuliert erhält eine Partei k Sitze, wenn der Quotient $\frac{v}{2 \cdot k - 1}$ zu den n_{seats} größten Höchstzahlen gehört, wobei v die Anzahl an Stimmen für die Partei bezeichnet. Ein Beispiel für die Anwendung des Höchstzahlverfahrens ist in Abbildung 4.1 a) zu sehen.

Das Divisorverfahren setzt voraus, dass ein geeigneter Divisor vorliegt. Durch diesen werden die Wählerstimmen pro Partei geteilt und anschließend werden die Quotienten nach Standardrundung gerundet. Bei Resten, die kleiner als $0,5$ sind, wird abgerundet und bei Resten, die größer als $0,5$ sind wird aufgerundet. Für einen Rest von genau $0,5$ ist keine eindeutige Rundung definiert. Das Ergebnis bildet pro Partei die Anzahl an Sitzen [15]. Ein Divisor ist genau dann geeignet, wenn die Summe der Sitze pro Partei genau der Gesamtanzahl an n_{seats} Sitzen entspricht. Im Allgemeinen gibt es mehrere geeignete Divisoren. Wenn kein Divisor bekannt ist, kann das Verfahren auch iterativ verwendet werden, indem mit einer Schätzung als Divisor angefangen wird und je nachdem,

a) Höchstzahlverfahren

	Partei A	Partei B	Partei C
$\frac{v_i}{1}$	100 (2)	150 (1)	90 (3)
$\frac{v_i}{3}$	33,3 (5)	50 (4)	30
$\frac{v_i}{5}$	20	30	18
$\frac{v_i}{7}$	14,3	21,4	12,9

b) Divisorverfahren mit Divisor d = 65

Partei	$\frac{v_i}{d}$	Sitze
Partei A	$\frac{100}{65} \approx 1,53$	2
Partei B	$\frac{150}{65} \approx 2,31$	2
Partei C	$\frac{90}{65} \approx 1,38$	1

Abbildung 4.1: Beispiel für das Sainte-Laguë-Verfahren: Partei A erhält 100 Stimmen, Partei B 150 Stimmen und Partei C 90 Stimmen und fünf Sitze sind zu verteilen. In a) wird das Höchstzahlverfahren angewandt. Daher werden die Höchstzahlen absteigend sortiert und die fünf größten markiert. Das Ergebnis ist, dass die Parteien A und B je zwei Sitze und Partei C einen Sitz bekommen. b) löst die Sitzverteilung äquivalent mit dem Divisor-Verfahren. Dabei werden die Stimmen der Parteien durch einen Divisor geteilt. Das kaufmännisch gerundete Ergebnis entspricht der Anzahl an Sitze für diese Partei. 65 ist für dieses Beispiel ein geeigneter Divisor, da die Summe an verteilten Sitzen genau der gewünschten Sitzanzahl entspricht.

ob die Summe der Sitze zu groß oder zu klein ist, der Divisor erhöht oder verringert wird [46]. In Abbildung 4.1 b) wird ein Beispiel für das Divisorverfahren angegeben. Auch hier zeigt sich, dass das Höchstzahl- und Divisorverfahren zum gleichen Ergebnis kommen.

4.8.2 Uneindeutigkeit bei der Sitzverteilung

Sowohl beim Höchstzahlverfahren als auch beim Divisorverfahren kann es vorkommen, dass das Ergebnis nicht eindeutig ist. Ein Beispiel hierfür ist in Abbildung 4.2 zu sehen. Auch wenn es unwahrscheinlich ist, dass dieser Fall eintritt, muss er aufgelöst werden, da es in der Praxis nicht akzeptabel ist, wenn mehr oder weniger Sitze vergeben werden, als im Parlament vorgesehen sind.

Beim Höchstzahlverfahren zeigt sich die Uneindeutigkeit dadurch, dass mehrere Höchstzahlen gleich sind, sodass keine eindeutige Sortierung durchgeführt werden kann. Eine Möglichkeit, dies aufzulösen, ist die Reihenfolge zwischen gleichen Höchstzahlen zufällig zu bestimmen.

Das Divisorverfahren zeigt eine Uneindeutigkeit dadurch, dass der Rest nach der Division mit einem geeigneten Divisor genau $0,5$ ist und es nur einen möglichen Divisor gibt [7]. Auch hier wird das Problem oft dadurch gelöst, dass bei einem Rest von $0,5$ zufällig auf- oder abgerundet wird [46].

Alle Ergebnisse, die durch die Auflösung entstehen, sind gleichwertig. Also sind beide Sitzverteilungen für das Beispiel in Abbildung 4.2 ein gültiges Ergebnis.

a) Höchstzahlverfahren

	Partei A	Partei B	Partei C
$\frac{v_i}{1}$	100 (2)	150 (1)	90 (3)
$\frac{v_i}{3}$	33,3 (5)	50 (4)	30 (6)
$\frac{v_i}{5}$	20	30 (6)	18
$\frac{v_i}{7}$	14,3	21,4	12,9

b) Divisorverfahren mit Divisor d = 60

Partei	$\frac{v_i}{d}$	Sitze
Partei A	$\frac{100}{60} \approx 1,67$	2
Partei B	$\frac{150}{60} = 2,5$	2 oder 3
Partei C	$\frac{90}{60} = 1,5$	1 oder 2

Abbildung 4.2: Beispiel für die Mehrdeutigkeit des Sainte-Laguë-Verfahrens: Wenn sechs Sitze zu verteilen sind, sind im Höchstzahlverfahren (a) die sechs größten Höchstzahlen nicht eindeutig. Beim Divisorverfahren (b) ist die Rundung bei einem Rest von $0,5$ nicht definiert. In beiden Fällen muss durch Zufall entschieden werden und beide Ergebnisse (2, 3, 1) und (2, 2, 2) sind valide.

Am Beispiel 4.2 lässt sich ein weiteres Problem bei der Auflösung mit dem Divisorverfahren erkennen: Um auf die vorgegebenen sechs Sitze zu kommen, muss bei Partei B auf- und bei Partei C abgerundet werden oder umgekehrt. Beim zufälligen Auf- bzw. Abrunden pro Partei darf also nicht der Fall eintreten, dass beide sich gleich verhalten. Daher darf der Zufall nicht unabhängig pro Partei stattfinden, sodass das Divisorverfahren bei einer Auswertung mit Tally Hiding nicht ohne weiteres umgesetzt werden kann, da sowohl der Divisor als auch die Stimmen als auch die zu vergebenen Sitze verschlüsselt sein müssen.

4.8.3 Modifiziertes Sainte-Laguë-Verfahren

In vielen skandinavischen Ländern kommt das modifizierte Sainte-Laguë-Verfahren zum Einsatz [15]. Dies funktioniert wie das Höchstzahlverfahren mit der Ausnahme, dass in der ersten Zeile der Tabelle die Stimmen pro Partei durch $1,4$ anstatt durch 1 geteilt werden. In den weiteren Zeilen wird die Reihe $3, 5, 7, \ldots$ wie beim klassischen Höchstzahlverfahren verwendet. Das modifizierte Höchstzahlverfahren hat die Eigenschaft, dass Parteien mehr Stimmen benötigen, um den ersten Sitz zu bekommen. Da die Modifizierung nur den ersten Sitz beeinflusst, ist das modifizierte Sainte-Laguë-Verfahren äquivalent zum klassischen Verfahren, wenn alle Parteien mindestens zwei Sitze bekommen.

4.8.4 Umwandlung des Divisorverfahrens ins Höchstzahlverfahren

Beide Verfahren sind äquivalent [46] und aus dem Höchstzahlverfahren lässt sich ein geeigneter Divisor berechnen [7]. Diese Eigenschaft wird in Kapitel 6 benötigt, um ein Höchstzahlverfahren mit gegebenem Divisor zu entwickeln.

Theorem 3 (Beziehung zwischen Divisor- und Höchstzahlverfahren)
Bei einer Verteilung von n_{seats} Sitzen nach Sainte-Laguë sind das Divisor- und das Höchstzahlverfahren äquivalent und es gilt folgendes: Jeder geeignete Divisor d aus dem Divisorverfahren liegt im Intervall $(2 \cdot h(n_{seats}+1), 2 \cdot h(n_{seats})]$ liegt, wobei $h(n_{seats})$ die n_{seats}-größte Höchstzahl bezeichnet.

Beweis Der Beweis ist auch in Quelle [7] beschrieben. Seien n_{seats} die Anzahl an zu verteilenden Sitzen und $s_1, ..., s_n$ die Sitze pro Partei, die durch das Sainte-Laguë-Verfahren berechnet werden. Nach dem Divisorverfahren werden diese folgendermaßen berechnet:

$$s_i = \left\lfloor \frac{v_i}{d} \right\rceil, \sum_{i=1}^{n} s_i = n_{seats}$$

Der erste Teil der Formel lässt sich wie folgt umformen:

$$s_i = \left\lfloor \frac{v_i}{d} \right\rceil \tag{4.1}$$

$$\iff s_i - \frac{1}{2} \leq \frac{v_i}{d} \leq s_i + \frac{1}{2} \tag{4.2}$$

$$\iff \frac{1}{s_i - \frac{1}{2}} \geq \frac{d}{v_i} \geq \frac{1}{s_i + \frac{1}{2}} \tag{4.3}$$

$$\iff \frac{2 \cdot v_i}{2 \cdot s_i - 1} \geq d \geq \frac{2 \cdot v_i}{2 \cdot s_i + 1} \tag{4.4}$$

Da diese Eigenschaft für alle Parteien i gilt und der Divisor unabhängig von der Partei ist, gilt insgesamt:

$$\min_{s_i} \frac{2 \cdot v_i}{2 \cdot s_i - 1} \geq d \geq \max_{s_i} \frac{2 \cdot v_i}{2 \cdot s_i + 1} \tag{4.5}$$

$$\implies \min_{s_i} \frac{2 \cdot v_i}{2 \cdot s_i - 1} \geq \max_{s_i} \frac{2 \cdot v_i}{2 \cdot s_i + 1} \tag{4.6}$$

Die letzte Zeile entspricht der Formulierung des Höchstzahlverfahrens. Aus dieser Umformung folgt, dass der Divisor im Intervall $(2 \cdot h(n_{seats}+1), 2 \cdot h(n_{seats})]$ liegt. ∎

4.9 Storting (Norwegen)

Das norwegische Parlament wird nach einer Verhältniswahl gewählt [35] und die Größe des Parlaments ist auf 169 Abgeordnete fixiert. Antreten können hier nicht nur Parteien, sondern auch andere politische Gruppen, solange diese korrekt angemeldet sind und mit einer Kandidatenliste in ihrem Bezirk antreten [26]. Jedem der 19 Wahlbezirke stehen eine festgelegte Anzahl an Sitzen zu und diese werden nach dem Verhältnis der Stimmen in diesem Bezirk verteilt [35]. Zusätzlich werden Ausgleichssitze verteilt, die die Zusammensetzung des Parlaments näher an die Gesamtverteilung der Stimmen angleichen sollen [35]. Da alle Parteien nur lokal mit Kandidatenlisten pro Bezirke antreten, werden die Ausgleichssitze anschließend wieder auf die Bezirke verteilt. Dabei wird in Kauf genommen, dass diese Verteilung nicht zu einem idealen Ergebnis kommen kann, dafür ist die Anzahl der Sitze im Parlament im Gegensatz zum Wahlsystem der Bundestagswahl fest.

Jeder Wähler gibt eine Stimme für eine Partei oder andere politische Gruppierung an, die in seinem Wahlbezirk antritt. Diese Stimme fließt sowohl in die Berechnung der Bezirkssitze als auch in die Berechnung der Ausgleichssitze ein [27].

Das Ergebnis ist die Anzahl der Sitze, die jeder Partei oder sonstiger Gruppe in jedem Wahlkreis zusteht. Davon kann öffentlich abgeleitet werden, welche Abgeordneten ins Parlament einziehen, denn dies geschieht nach dem Listenplatz jedes Kandidaten.

In den folgenden Abschnitten wird die Auswertung einer Wahl des norwegischen Parlaments genau beschrieben. Eine Übersicht über die einzelnen Schritte ist in Abbildung 4.3 dargestellt.

4.9.1 Verteilung der Bezirkssitze

Jedem Bezirk stehen eine vor der Wahl festgelegte Anzahl an Sitzen zu. Diese Anzahl ist abhängig von der Einwohnerzahl und der Fläche des Bezirks [26]. Obwohl diese Verteilung alle acht Jahre aktualisiert wird, wird sie bei der offiziellen Auswertung [27] als gegeben angenommen, weshalb sie auch für diese Arbeit nicht berechnet wird. Die Gesamtanzahl von in Summe 150 Bezirkssitzen ändert sich nicht.

Innerhalb eines Bezirks werden die Sitze nach dem modifizierten Sainte-Laguë-Verfahren verteilt, wie es in Abschnitt 4.8.3 beschrieben wurde. Wenn es dabei zwischen mehreren Parteien zu einem Gleichstand kommt, wird die Partei bevorzugt, die in dem Wahlbezirk mehr Stimmen erhalten hat. Wenn auch diese gleich ist, wird durch Los entschieden [4]. Es ist garantiert, dass jede Partei mindestens so viele Sitze erhält, wie ihr Bezirkssitze zustehen [26].

4.9.2 Verteilung der Ausgleichssitze

Vor der Verteilung der 19 Ausgleichssitze müssen die Stimmen aus den Wahlbezirken landesweit pro Partei aufsummiert werden. Nur echten Parteien, die mindestens vier Prozent aller landesweit abgegebenen Stimmen erhalten, können Ausgleichssitze zugewiesen werden [4]. Nun wird die Sainte-Laguë-Menge definiert. Dies Menge enthält genau die Parteien, die Ausgleichsmandate erhalten und ist initial die Menge aller Parteien über der 4%-Hürde. Weiter ist n_{seats} die Anzahl der zu verteilenden Sitze. Initial wird n_{seats} auf die 169 Gesamtsitze des Parlaments gesetzt, aber die Anzahl an Bezirkssitze, die von Parteien unterhalb der 4%-Hürde erhalten werden, wird von n_{seats} abgezogen.

Im ersten Schritt werden die n_{seats} Sitze nach dem modifizierten Sainte-Laguë-Verfahren auf die Parteien der Sainte-Laguë-Menge verteilt. Anschließend wird für jede dieser Parteien die Anzahl der Sitze nach dem Sitzzuteilungsverfahren mit der Anzahl der Bezirkssitze verglichen. Wenn einer Partei mehr Bezirkssitze zustehen, als sie nach dem Sainte-Laguë-Verfahren erhält, wird diese Partei aus der Sainte-Laguë-Menge entfernt und die Anzahl an zu verteilenden Sitzen n_{seats} wird um die Anzahl der Bezirkssitze dieser Partei reduziert. Dieser Schritt wird iterativ so oft wiederholt, bis keine Partei der Sainte-Laguë-Menge mehr Bezirkssitze als Sitze nach dem Sitzzuteilungsverfahren hat [4].

Somit steht die Anzahl an Sitzen pro Partei fest: Jede Partei der Sainte-Laguë-Menge erhält so viele Sitze, wie ihr nach Berechnung des modifizierten Sainte-Laguë-Verfahrens in der letzten Iteration zustehen und die restlichen Parteien erhalten jeweils ihre Bezirkssitze. In Summe wurden also 169

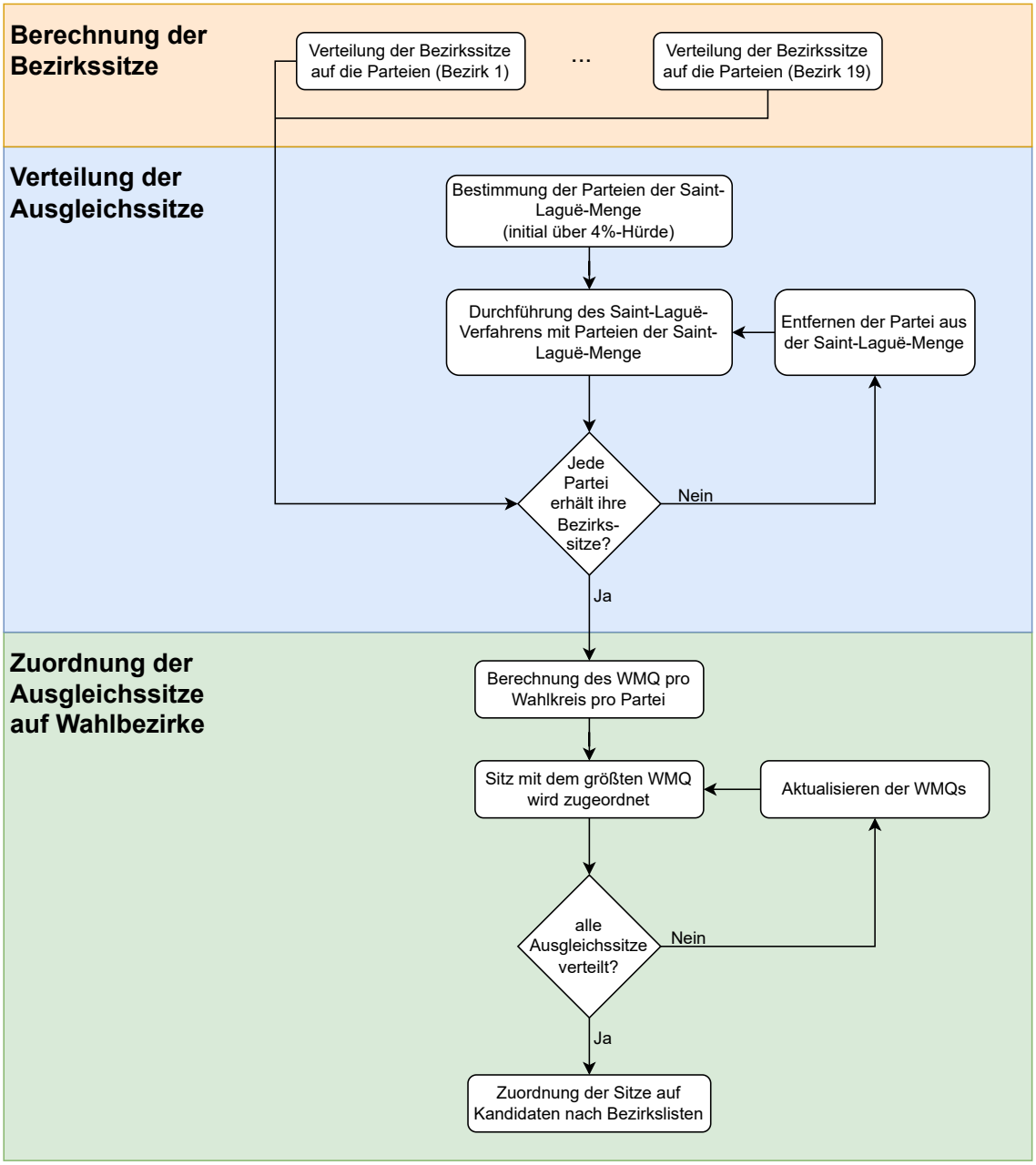


Abbildung 4.3: Überblick über das norwegische Wahlverfahren: Im ersten Schritt werden in jedem Wahlbezirk die Bezirkssitze verteilt. Anschließend werden iterativ Ausgleichssitze an Parteien der Sainte-Laguë-Menge vergeben; diese werden wiederum auf die Wahlbezirke verteilt. (Die Abkürzung WMQ steht für den gewichteten Mandatsquotienten.)

Sitze auf die Parteien verteilt, und jeder Bezirkssitz ist einem Wahlbezirk zugeordnet, aber die 19 Ausgleichssitze wurden nur landesweit verteilt und sind noch keinem Bezirk zugeordnet. Diese Zuordnung wird im letzten Schritt durchgeführt.

Zuordnung der Ausgleichssitze auf die Wahlbezirke

Die 19 Ausgleichssitze der Parteien der Sainte-Laguë-Menge werden so auf die Wahlbezirke verteilt, dass jeder Wahlbezirk genau einen Ausgleichssitz erhält und dieses Mandat bestmöglich zur lokalen Stimmverteilung des Bezirks passt. Auch die Verteilung der Ausgleichssitze findet iterativ statt [4]:

Zuerst wird für jede Partei p, die insgesamt mindestens einen Sitz erhalten hat, und für jeden Wahlbezirk b, in dem diese Partei Stimmen bekommen hat der folgende Mandatsquotient berechnet [4]:

$$\text{MQ}(p,b) = \frac{v_{p,b}}{2 \cdot s_{p,b} + 1} \tag{4.7}$$

Dabei bezeichnen $v_{p,b}$ die Anzahl an Stimmen, die Partei p in Wahlbezirk b erhalten hat und $s_{p,b}$ die Anzahl an Sitzen, die p in b erhalten hat.

Daraus wird der gewichtete Mandatsquotient berechnet [4]:

$$WMQ(p,b) = \begin{cases} MQ(p,b) \cdot \frac{s_b}{v_b} & \text{, p hat Überhang und b einen freien Ausgleichssitz} \\ 0 & \text{, sonst} \end{cases} \tag{4.8}$$

Hier bezeichnen s_b die Gesamtanzahl an Sitzen in Bezirk b und v_b die Anzahl an gültigen Stimmen, die in b abgegeben wurden. Eine Partei hat Überhang, wenn sie mindestens einen Ausgleichssitz hat, der keinem Bezirk zugeteilt ist.

Anschließend werden die gewichteten Mandatsquotienten der Große nach sortiert. Wenn mehrere Quotienten gleich groß sind, wird der Quotient, in dessen zugehörigen Bezirk mehr Wähler ihre Stimme abgegeben haben, als größer angesehen. Sollte dann noch immer Gleichstand herrschen, wird durch Zufall entschieden [4].

Nun wird der größte gewichtete Mandatsquotient $WMQ(\hat{p}, \hat{b})$bestimmt und ein Ausgleichssitz der Partei $\hat{p}$ wird dem Wahlbezirk $\hat{b}$ zugeordnet. Anschließend werden alle Quotienten erneut berechnet und der Zuordnungsschritt wiederholt. Dies wird so oft durchgeführt, bis alle Ausgleichsmandate einem Bezirk zugeordnet sind [4].

4.10 Deutscher Bundestag

Die deutsche Bundestagswahl ist eine personalisierte Verhältniswahl, das heißt, sie setzt sich aus einer Personenwahl über die Erststimmen und einer Verhältniswahl über die Zweitstimmen zusammen. Dabei wird über die Erststimme ein Kandidat im Wahlkreis gewählt und von jedem Wahlkreis zieht der Kandidat mit den meisten Stimmen als Direktmandat in den Bundestag ein. Mit der Zweitstimme wird eine Partei gewählt und die Sitzverteilung der Parteien untereinander wird durch

mehrmaliges Anwenden des Sainte-Laguë-Verfahrens ermittelt. Die Evaluation der Bundestagswahl legt großen Wert darauf, dass durch die Direktmandate das nach Zweitstimmen berechnete Verhältnis der Parteien untereinander nur minimal gestört wird. Daher können zusätzliche Mandate, sogenannte Ausgleichsmandate, vergeben werden, die das korrekte Verhältnis wiederherstellen [46]. Bei Gleichstand wird im Allgemeinen durch Zufall entschieden [46]. Ein Schema für die gesamte Auswertung ist in Abbildung 4.4 dargestellt und im Folgenden wird die Auswertung detailliert beschrieben.

4.10.1 Ergebnis

Das Ergebnis der Bundestagswahl ist eine Sitzverteilung mit mindestens 598 Sitzen. Pro Sitz wird dabei die Person angegeben, die dieses Mandat erhält. Es wird nicht unterschieden, ob einem Abgeordneten das Mandat durch ein Direktmandat oder durch einen Listenplatz zusteht. Diese Information wird bewusst verschlüsselt gehalten und ist nicht Teil des Ergebnisses, denn in der Realität sehen sich Abgeordnete auch dann als Vertreter ihres Wahlkreises, wenn sie über einen Listenplatz in den Bundestag eingezogen sind.

4.10.2 Auswertung der Erststimmen

Pro Wahlkreis wird ein Abgeordneter durch eine einfache Mehrheitswahl gewählt, das heißt der Kandidat mit den meisten Stimmen zieht als Direktmandat in den Bundestag ein. Bei einem Gleichstand der Stimmen zwischen zwei oder mehr Kandidaten wird durch Zufall entschieden [9].

4.10.3 Erste Verteilungsstufe

Die Gesamtauswertung wird in zwei Verteilungsstufen aufgeteilt [46]. Ziel der ersten Stufe ist es, für jede Partei einen sogenannten Mindestsitzanspruch zu berechnen [46]. Diese vorläufige Sitzverteilung gibt an, wie viele Sitze jeder Partei mindestens zustehen. Dabei wird immer von den vorgesehenen 598 Sitzen ausgegangen. In der zweiten Stufe wird die Sitzzahl so lange erhöht, bis jeder Mindestsitzanspruch erfüllt ist und die Eigenschaften der Verhältniswahl ebenfalls erfüllt sind.

Jede Oberverteilung führt die Berechnungen auf Bundesebene durch, während die Unterverteilungen auf Länderebene berechnet werden.

Erste Oberverteilung

Die erste Oberverteilung verteilt die Gesamtanzahl von 598 Sitzen auf die einzelnen Bundesländer. Dazu wird das Verfahren von Sainte-Laguë verwendet. Doch anstatt wie beim klassischen Sainte-Laguë-Verfahren die Sitze nach Stimmen zu verteilen, wird für die Verteilung die Anzahl der Einwohner pro Bundesland verwendet [46].

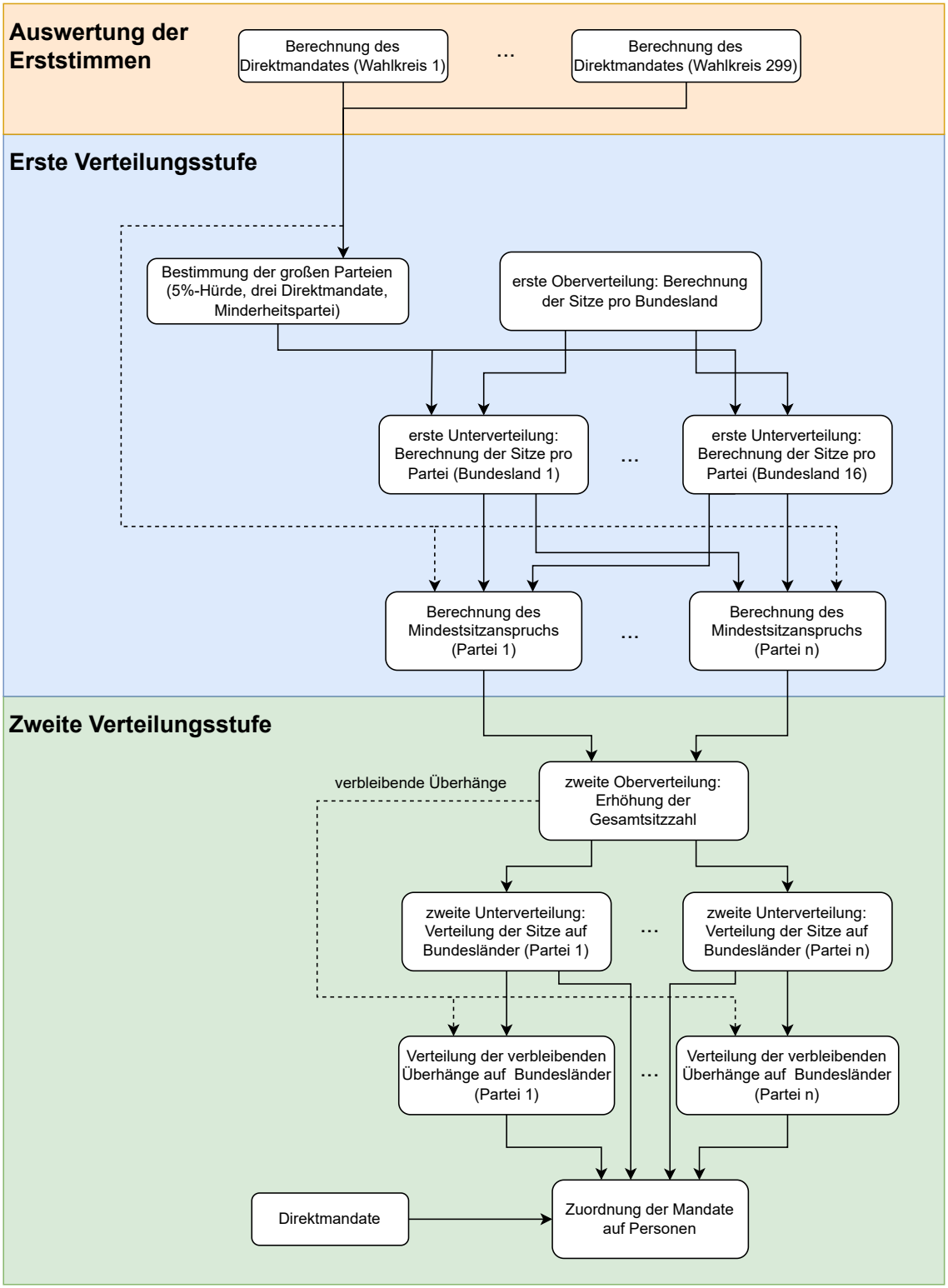


Abbildung 4.4: Überblick über das Wahlverfahren der Bundestagswahl: Zuerst werden die Direktmandate berechnet. Anschließend wird in der ersten Verteilungsstufe der Mindestsitzanspruch pro Partei bestimmt. In der zweiten Verteilungsstufe wird die Gesamtsitzzahl erhöht, um Überhangmandate auszugleichen und die Sitze werden auf die Parteien und auf die Bundesländer verteilt.

Erste Unterverteilung

Ziel der ersten Unterverteilung ist es, die berechneten Sitze pro Bundesland auf die einzelnen Parteien, die in dem Bundesland angetreten sind, zu verteilen [46]. Ab diesem Berechnungsschritt werden für die gesamte Auswertung nur noch große Parteien betrachtet. Eine Partei ist groß, wenn sie mindestens 5% der Zweitstimmen oder mindestens drei Direktmandate erhalten hat oder eine Minderheitspartei ist.

Bei der Unterverteilung wird pro Bundesland erneut das Sainte-Laguë-Verfahren angewandt. In jedem Bundesland werden so viele Sitze verteilt, wie in der ersten Oberverteilung berechnet werden und für die Verteilung sind die Zweitstimmen des jeweiligen Bundeslandes relevant. Somit steht für jedes Bundesland und jede Partei fest, wie viele Sitze einer Partei in einem Bundesland nach den Zweitstimmen zustehen. Diese Zahl wird als Sitzkontingent bezeichnet. Im nächsten Schritt werden auch die Erststimmen mit einbezogen.

Berechnung des Mindestsitzanspruchs

In diesem Berechnungsschritt wird berechnet, wie viele Sitze einer Partei mindestens zustehen, abhängig davon wie viele Sitze sie durch die erste Unterverteilung erhalten hat und wie viele Direktmandate sie gewonnen hat. Dafür wird für jede Partei folgendes berechnet: Für jedes Bundesland werden das Sitzkontingent und die Anzahl der Direktmandate in dem Bundesland benötigt. Daraus wird zuerst der drohende Überhang berechnet. Dieser ist null, wenn eine Partei ein größeres Sitzkontingent als Direktmandate bekommt. Ansonsten ist der drohende Überhang die Differenz aus Direktmandaten und dem Sitzkontingent [46].

Anschließend wird die sogenannte Mindestsitzzahl pro Partei berechnet [46]:

$$s_i^{min} = \max\left\{\left\lceil \frac{s_i^{dir} + s_i^{kon}}{2} \right\rceil, s_i^{dir}\right\} \tag{4.9}$$

Nun werden für jede Partei sowohl die Mindestsitzzahlen als auch die Sitzkontingente über jedes Bundesland aufsummiert. Das Maximum dieser beiden Summen wird als Mindestsitzanspruch der Partei bezeichnet [46].

4.10.4 Zweite Verteilungsstufe

In dieser Stufe werden die endgültige Sitzverteilung und das Ergebnis berechnet, indem die Gesamtanzahl der Sitze so lange erhöht werden, bis alle Bedingungen erfüllt sind.

Zweite Oberverteilung

In der zweiten Oberverteilung wird für jede Partei der Mindestsitzanspruch und der gesamte drohende Überhang verwendet. Ziel ist es, die finale Sitzzahl des Bundestags und der einzelnen Parteien zu berechnen. Dabei werden im Allgemeinen 598 Sitze als Ausgangssituation verwendet. Wenn es Direktmandate von kleinen Parteien gibt, werden diese von den 598 Sitzen abgezogen.

Konzeptionell wird nun die Sitzzahl so lange erhöht, bis durch das Sainte-Laguë-Verfahren eine Verteilung entsteht, in der jede Partei ihren Mindestsitzanspruch erhält. Für insgesamt bis zu drei Überhänge kann dabei eine Ausnahme gemacht werden und die jeweiligen Parteien müssen nur ihre Mindestsitzzahl minus die unausgeglichenen Überhänge erreichen. In diesem Schritt wird das Tie-Breaking nicht durch Zufall durchgeführt, sondern bei Gleichstand wird allen Parteien ein Sitz zugeordnet. Dadurch werden bei Gleichstand mehr Sitze vergeben.

Anstatt die Sitze inkrementell zu erhöhen, wird vom Bundeswahlleiter ein effizienterer Algorithmus beschrieben [46]: Ziel ist es, einen Divisor für das Sainte-Laguë-Verfahren aufzustellen, für den alle Bedingungen erfüllt sind. Dies besteht aus zwei Teilschritten, in denen ein Divisor ohne Überhänge und ein Divisor mit Überhängen berechnet werden. Der Divisor ohne Überhänge wird mit den in Abschnitt 4.10.3 berechneten Mindestsitzansprüchen folgendermaßen bestimmt:

$$d_{ohne} = \min_{i \in \text{große Parteien}} \left\{ \frac{v_i}{s_i^{min} - 0,5} \right\} \tag{4.10}$$

Dabei bezeichnen v_i die Zweitstimmen für Partei i und s_i^{min} den Mindestsitzanspruch.

Aus den Parteien, die drohende Überhänge haben, wird die folgende Menge definiert:

$$D = \underbrace{\left\{ \frac{v_i}{s_i^{min} - 0,5} \right\} \cup \left\{ \frac{v_i}{s_i^{min} - 1,5} \right\}}_{\text{wenn Überhang (von Partei i)} \geq 1} \cup \underbrace{\left\{ \frac{v_i}{s_i^{min} - 2,5} \right\}}_{\text{wenn Überhang (von i)} \geq 2} \cup \underbrace{\left\{ \frac{v_i}{s_i^{min} - 3,5} \right\}}_{\text{wenn Überhang (von i)} \geq 3} \tag{4.11}$$

Das viertkleinste Element von D ist der Divisor mit Überhängen. Hier wird nicht das kleinste Element gewählt, da bis zu drei Überhänge unausgeglichen sein dürfen.

Der gesuchte Divisor für das Sainte-Laguë-Verfahren ist das Minimum der beiden berechneten Hilfsdivisoren. Mit diesem Divisor wird die Sitzverteilung so berechnet, dass die Bedingungen erfüllt sind, dass jede Partei ihren Mindestsitzanspruch erhält und es maximal drei unausgeglichene Überhangmandate gibt [46].

Zweite Unterverteilung

In der zweiten Unterverteilung werden die in der zweiten Oberverteilung berechneten Gesamtsitze pro Partei auf die einzelnen Bundesländer verteilt, da die Parteien ihre Kandidaten in Landeslisten, nicht in einer Bundesliste, aufstellen. Dabei wird nochmal das Sainte-Laguë-Verfahren mit der erhöhten Gesamtsitzzahl angewandt. Damit sind alle Sitze außer die drei unausgeglichenen Überhangmandate auf die einzelnen Parteien und innerhalb einer Partei auf die einzelnen Bundesländer verteilt.

Verteilung der unausgeglichenen Überhänge

Für die Parteien, die unausgeglichene Überhänge erhalten haben, müssen diese noch auf eine oder mehrere Landeslisten aufgeteilt werden. Analog zu Abschnitt 4.10.4 wird für jede Partei i mit unausgeglichen Überhängen die folgende Menge definiert, wobei l über alle Bundesländer iteriert:

(4.12)
$$L_i = \left\{ \frac{v_{i,l}}{s_{i,l}^{min} - 0,5} \right\} \cup \underbrace{\left\{ \frac{v_{i,l}}{s_{i,l}^{min} - 1,5} \right\}}_{\text{Überhänge (von i)} \geq 2} \cup \underbrace{\left\{ \frac{v_{i,l}}{s_{i,l}^{min} - 2,5} \right\}}_{\text{Überhänge (von i)} \geq 3}$$

Das erste unausgeglichene Überhangmandat wird dem Bundesland mit dem kleinsten Element in L_i zugeordnet, wenn die Partei noch ein zweites unausgeglichenes Überhangmandat hat, wird das dem Bundesland mit dem zweitkleinsten Element in L_i zugeordnet, usw. [46].

Auswertung des Ergebnisses

Das Ergebnis wird pro Partei und pro Bundesland berechnet. Zuerst erhalten die Kandidaten der Landesliste ein Mandat, die in einem Wahlkreis des Bundeslandes direkt gewählt wurden. Anschließend wird die Landesliste von oben aufgefüllt, bis die berechnete Sitzzahl erreicht ist [46].

5 Allgemeine Änderungen bei der Umsetzung politischer Wahlverfahren aus der Praxis

Ziel dieser Bachelorarbeit ist es, Ordinos um Wahlverfahren, wie sie in der Praxis angewandt werden, zu erweitern. Der Unterschied zwischen den durch frühere Arbeiten umgesetzten generischen Wahlverfahren und Wahlverfahren aus der Praxis ist, dass generische Wahlverfahren von Details abstrahieren und daher als Vorlage für ähnliche praxisnahe Wahlen verwendet werden können. Bei Wahlverfahren aus der Praxis hingegen wird die Auswertung der Wahl genau so implementiert, wie sie in der Realität ausgeführt wird und die Umsetzung beachtet dabei alle Details und Sonderregelungen der entsprechenden Wahl.

In diesem Kapitel werden die allgemeinen Änderungen beschrieben, die beim Übergang von generischen Wahlverfahren zu politischen Wahlverfahren aus der Praxis anfallen und es wird auf die Besonderheiten der in dieser Arbeit hinzugefügten Wahlverfahren eingegangen. Dies beinhaltet eine Konkretisierung der Wahlsysteme wie das Wählen in Wahlkreisen und unterschiedliche Arten von Stimmzetteln, das Beachten von Randfällen, die in allgemeinen Verfahren vernachlässigt werden, wie das Verhalten bei Gleichstand sowie Überlegungen für die Laufzeit und Effizienz der Verfahren.

In Kapitel 6 wird an drei verschiedenen Wahlsystemen beispielhaft gezeigt, wie solche Wahlen in Ordinos umgesetzt werden können.

5.1 Wählen in Wahlkreisen

Bei vielen politischen Wahlen wird in unterschiedlichen Wahlkreisen gewählt, und in welchem Wahlkreis eine Stimme abgegeben wird, wirkt sich darauf aus, wo sie gezählt wird. Daher müssen alle Stimmzettel um die jeweiligen Wahlkreise ergänzt werden. Der Wahlkreis auf dem Stimmzettel kann entweder verschlüsselt oder unverschlüsselt auf das Bulletin Board geschrieben werden. Unverschlüsselte Wahlkreise haben den Vorteil, dass die Stimmaggregation schneller ist, da die Stimmzettel nach ihrem Wahlkreis sortiert und dadurch nur innerhalb des Wahlkreises aggregiert werden müssen. In Bezug auf die Vote Privacy gibt es keinen Unterschied, ob der Wahlkreis eines Wählers unverschlüsselt oder verschlüsselt vorliegt, da Wahlkreise bekannte Informationen sind. Der Angreifer kennt insbesondere den Wahlkreis des Wählers unter Beobachtung, denn wenn in der Realität ein Wähler beobachtet wird, ist dem Angreifer der Wohnort des Wählers bekannt und dadurch lässt sich der Wahlkreis ableiten.

Daher wurde Ordinos um modifizierte Stimmzettel ergänzt, die neben der verschlüsselten Stimmabgabe des Wählers auch den Wahlkreis beinhalten.

Auch die Aggregation wurde angepasst: Wenn die Trustees die abgegebenen Stimmen auf ihre Gültigkeit überprüfen, werden sie zuerst nach Wahlkreisen sortiert. Das für die Wahl definierte Aggregationsprotokoll bezieht sich nicht mehr auf alle Stimmen, sondern gilt pro Wahlkreis, da in allen umgesetzten Wahlen die Aggregation in jedem Wahlkreis identisch abläuft. Also werden die Stimmen pro Wahlkreis lokal aggregiert und anschließend an das Evaluierungsprotokoll weitergegeben.

Indem die Stimmen nach Wahlkreis getrennt aggregiert werden, wird eine Änderung am allgemeinen Ordinos-Protokoll durchgeführt. Daher muss gezeigt werden, dass die drei Sicherheitseigenschaften Tally Hiding, Vote Privacy und Accountability auch nach dieser Änderung erfüllt sind.

Theorem 4 (Sicherheit der nach Wahlkreis getrennten Aggregation)
Sei P_{eval} ein MPC-Protokoll für Ordinos, das eine beliebige Wahl mit Wahlkreisen auswertet und die Sicherheitseigenschaften erfüllt. Weiter sei P_{agg} das im letzten Abschnitt beschriebene Aggregationsprotokoll nach Wahlkreisen. Kombiniert wird beides zu dem Protokoll $P_{ges} = P_{agg}||P_{eval}$. Dann gilt, dass P_{ges} sicher ist unter der Bedingung, dass P_{eval} sicher ist.

Beweis Das Protokoll P_{ges} ist sicher, wenn alle Voraussetzungen aus Theorem 1 und Theorem 2 erfüllt sind. Die Voraussetzungen (a) - (d) aus Theorem 1 für das Verschlüsselungsschema, den Signierungsalgorithmus sowie die Protokolle für die Schlüsselgenerierung und Verschlüsselung sowie Voraussetzung (a) aus Theorem 2 für die Ehrlichkeit gewisser Instanzen trivialerweise sind erfüllt, da sie bereits vorher in Ordinos erfüllt waren und nicht geändert wurden.

Für P_{ges} sind Tally Hiding und Vote Privacy gegeben: P_{agg} erhält als Input die Chiffretexte, die die Stimmen der einzelnen Wähler darstellen. Bei der Sortierung nach Wahlkreis wird nichts entschlüsselt und die Aggregationsprotokolle pro Wahlkreis erfüllen bereits alle Sicherheitseigenschaften [31]. Der Output von P_{agg} sind Chiffretexte, die verschlüsselt an P_{eval} übergeben werden. Dieses Protokoll ist nach Annahme sicher. Also wird durch die neue Aggregierung nach Wahlkreisen im Gesamtprotokoll kein Chiffretext entschlüsselt, der nicht das Ergebnis von P_{eval} ist. Somit ist auch Voraussetzung (e) aus Theorem 1 erfüllt, weshalb die beiden Sicherheitseigenschaften Tally Hiding und Vote Privacy eingehalten werden.

Auch die Accountability ist erfüllt, denn Voraussetzung (b), die besagt, dass erkannt wird, wenn der Output des Protokolls nicht das korrekte Ergebnis des Inputs ist, gilt: Die einzelnen verschlüsselten Stimmen stehen auf dem Bulletin Board, sodass auffällt, wenn ein Trustee mit anderen Werten als Input rechnet. Die Sortierung nach Wahlkreisen in P_{agg} ist öffentlich, da die Wahlkreise unverschlüsselt sind. Also kann durch die Dokumentation der Rechnungen der Trustees auf dem Bulletin Board überprüft werden, dass jeder Trustee richtig sortiert. Für jeden Wahlkreis ist also der Input bekannt und da die Aggregation für einen Wahlkreis ein sicheres Multi-Party-Computation-Protokoll ist [31], kann überprüft werden, dass der Output jedes Trustees zum Input passt. Der Gesamtoutput von P_{agg} steht wiederum öffentlich auf dem Bulletin Board, sodass kein Trustee unbemerkt etwas anderes als Input für P_{eval} verwenden kann, da P_{eval} nach Annahme sicher ist. ■

5.2 Unterschiedliche Kandidaten pro Wahlkreis

Eine weitere Herausforderung der Wahlkreise ist, dass nicht in jedem Wahlkreis die gleichen Kandidaten beziehungsweise Parteien antreten. Dabei gibt es in den vorgestellten Wahlen zwei Fälle: Im ersten Fall werden Kandidaten nur für einen Wahlkreis gewählt. Ein Beispiel hierfür ist

die Wahl des britischen House of Commons. Dabei wird in jedem Wahlkreis aus unterschiedlichen Kandidaten ein Abgeordneter gewählt, der dann ins Parlament einzieht [18]. Die Stimmen, die die anderen Kandidaten erhalten, sind nach Feststellung des Siegers nicht mehr relevant. Somit ist auch irrelevant, welche Kandidaten in anderen Wahlkreisen angetreten sind. Da ein Kandidat maximal für einen Wahlkreis antreten kann, sind die Auswertungen der einzelnen Wahlkreise gänzlich voneinander getrennt.

Daher wurde sich in diesem Fall dafür entschieden, auch die Kandidaten wahlkreisspezifisch in den Stimmzetteln darzustellen, sodass genau die Kandidaten auf dem Stimmzettel eines Wahlkreises stehen, die in diesem Wahlkreis antreten. Daraus folgt, dass Stimmabgaben von Wahlkreis zu Wahlkreis unterschiedlich interpretiert werden müssen. Als Beispiel stehen in Wahlkreis 1 die Parteien A und B zur Wahl und in Wahlkreis 2 die Parteien A und C. Der Stimmzettel $(\mathrm{E_{pk}}(0), \mathrm{E_{pk}}(1))_{\text{Wahlkreis 1}}$ wird so interpretiert, dass in Wahlkreis 1 eine Stimme für Partei B abgegeben wird. Hingegen besagt der Stimmzettel $(\mathrm{E_{pk}}(0), \mathrm{E_{pk}}(1))_{\text{Wahlkreis 2}}$, dass in Wahlkreis 2 eine Stimme für Partei C abgegeben wird. Diese unterschiedliche Interpretation wird erst bei der Entschlüsselung am Ende der Evaluation aufgehoben. Wenn in Wahlkreis 2 die zweite zur Wahl stehende Partei gewonnen hat, wird vom Bulletin Board gelesen, welcher Partei dies entspricht und Partei C als Wahlkreisgewinner ausgegeben.

Im zweiten Fall werden Parteien zwar in einem Wahlkreis gewählt, aber auch Stimmen für unterlegene Parteien sind für das finale Ergebnis relevant. Ein Beispiel hierfür ist die Zweitstimme der Bundestagswahl. Dabei stehen pro Bundesland nicht die gleichen Parteien zur Wahl, da manche Parteien nur in einem oder wenigen Bundesländern antreten. Für die Bundestagswahl ist es in der Auswertung aber sowohl relevant, wie viele Stimmen eine Partei bundesweit erhält (z. B. für die Berechnung des Mindestsitzanspruchs) als auch wie viele Stimmen eine Partei in einem Bundesland gewinnt (z. B. für die Unterverteilung).

Es wurden drei Möglichkeiten in Betracht gezogen, um einerseits sicherzustellen, dass in einem Bundesland nur Parteien, die antreten, auch gewählt werden können und andererseits eine bundesweite Stimmaggregation zu ermöglichen: Zunächst könnte der Wähler dafür sorgen, dass er keine unwählbare Partei wählt. Dann wären auf dem Stimmzettel aus jedem Bundesland alle Parteien, die bundesweit antreten, aufgelistet, aber der Stimmzettel wäre nur gültig, wenn die Parteien, die in einem Bundesland nicht wählbar sind, als Stimme eine verschlüsselte Null zugewiesen bekommen. Dafür müsste der Wähler bei diesen Parteien einen Zero-Knowledge-Beweis angeben, dass es sich um eine Null handelt. Obwohl eine solcher Beweis in Ordinos bereits existiert, ist die Überprüfung teuer, weshalb sich dagegen entschieden wurde.

Eine andere Möglichkeit wäre, das Problem in den Authentication Server zu verlagern. Dann enthält der Stimmzettel, den der Wähler abgibt, nur die im Bundesland wählbaren Parteien und der Authentication Server ergänzt den Stimmzettel um die restlichen Parteien, bevor er den Stimmzettel auf das Bulletin Board schreibt. Dabei wird die Stimme der zusätzlichen Parteien auf Null gesetzt und die Zero-Knowledge-Beweise entfallen. Dennoch ist auch diese Methode nicht optimal. Es gab beispielsweise bei der Bundestagswahl 2021 mindestens 16 Parteien, die nicht in Baden-Württemberg angetreten sind, und fast 6 Millionen gültige Zweitstimmen wurden in Baden-Württemberg abgegeben [46]. Daraus folgt, dass bei einer Aggregation nach dem obigen Vorschlag für dieses Bundesland über 90 Millionen Mal unnötigerweise eine Addition stattfindet, von der schon vor der Wahl bekannt ist, dass das Ergebnis Null sein wird. Zusätzlich wird bei der Auswertung pro Bundesland über 16 Parteien iteriert und Berechnungen durchgeführt, bei denen bekannt ist, dass sie in diesem Bundesland keine Stimme erhalten können.

Aufgrund der hohen Wähler- und Parteizahlen bei großen politischen Wahlen, wurde sich dazu entschieden, die Anpassung an alle bundesweit antretenden Parteien erst so spät wie möglich in der Evaluation vorzunehmen. Das heißt, dass jede abgegebene Stimme nur die wählbaren Parteien beinhaltet und die Stimmen in diesem Format aggregiert und auszuwerten werden. Erst wenn die Auswertung bundesweite Aspekte enthält, werden in die Summen der Stimmen pro Partei die nicht wählbaren Parteien mit Null Gesamtstimmen eingefügt.

5.3 Unterschiedliche Arten von Stimmzetteln

Bei der Bundestagswahl kommt es zu dem Fall, dass es unterschiedliche Arten von Stimmzetteln gibt, da sowohl Erststimmen als auch Zweitstimmen abgegeben werden müssen [46]. Da in der Realität auch nur eine der beiden Stimmen gültig abgegeben werden kann [46], werden Erst- und Zweitstimme unabhängig voneinander modelliert, sodass für jeden Wähler zwei aktuelle gültige Stimmen auf dem Bulletin Board liegen können. Umgesetzt wurde dies, indem jeder Stimme noch ein Typ ”primary vote” bzw. ”secondary vote” hinzugefügt werden kann. Die Trennung zwischen Erst- und Zweitstimme ist auch insofern konsistent, da die jeweiligen Wahlkreise unterschiedlich sind. Bei der Erststimme muss der angegebene Wahlkreis dem tatsächlichen Wahlkreis des Wählers entsprechen. Doch bei der Zweitstimme wird pro Bundesland aggregiert. Daher wird bei der Zweitstimme statt dem Wahlkreis das Bundesland angegeben. Die Vote Privacy wird dadurch nicht nennenswert verbessert, da wie oben beschrieben der Wahlkreis des Wählers unter Beobachtung bekannt ist, aber die Aggregation ist einfacher gestaltet, da auch in diesem Fall nach der Variablen ”Wahlkreis” aggregiert wird, auch wenn im Wahlkreis ein Bundesland eingetragen ist.

5.4 Verhalten bei Gleichstand

Im Gegensatz zu bisher implementierten Wahlverfahren, muss bei in der Praxis verwendeten Wahlverfahren auch der Fall beachtet werden, dass es zu einem Gleichstand kommen kann. Dieser muss im Allgemeinen aufgelöst werden, beispielsweise um eine vorgegebene Sitzzahl nicht zu über- oder unterschreiten, aber bei der Auflösung des Gleichstandes dürfen die Sicherheitseigenschaft, insbesondere das Tally Hiding, nicht verletzt werden. Im Kontext von Wahlen wird das Auslösen eines Gleichstandes auch als Tie-Breaking bezeichnet.

In der Praxis werden unterschiedliche Verfahren zum Auflösen eines Gleichstandes zwischen zwei oder mehreren Kandidaten verwendet: Häufig wird durch Zufall entschieden, aber je nach Wahl kann es auch durch das Alter der Kandidaten oder die Position auf dem Wahlzettel aufgelöst werden. Zusätzlich gibt es die Möglichkeit, dass externe Personen oder Organisationen wie der Präsident, der Wahlvorstand oder eine Gruppe von Abgeordneten der Wahlbezirke entscheiden, wer Sieger der Wahl ist. Vor allem bei kleineren Wahlen kann den betroffenen Kandidaten die Möglichkeit gegeben werden, das Ergebnis unter sich auszuhandeln. Manche Wahlen werden durch einen Gleichstand als ungültig erklärt, sodass Neuwahlen stattfinden müssen und auch einige größere politische Wahlen haben offiziell keine Strategie für den Fall eines Gleichstandes festgelegt [39].

5.4.1 Vorbereitung für das Tie-Breaking

Wenn das Auflösen eines Gleichstandes erst beim Auftreten des Unentschiedens in Angriff genommen wird, müssen Informationen entschlüsselt werden, die nach dem Prinzip des Tally Hiding geheim bleiben sollen: Wenn beispielsweise bei einer Mehrheitswahl zwischen den beiden Kandidaten mit den meisten Stimmen ein Gleichstand herrscht, ist die naive Vorgehensweise, diese Information zu entschlüsseln, damit der Gleichstand aufgehoben werden kann. Doch dies führt dazu, dass ebenfalls bekannt ist, wer Zweiter geworden ist, wodurch das Tally Hiding verletzt wird. Eine Verbesserung dazu wäre, nur preiszugeben, dass ein Gleichstand herrscht, aber nicht zwischen welchen Kandidaten. Aber bei Wahlen mit wenigen Wählern und wenigen Kandidaten können durch das Wissen, dass mindestens zwei Kandidaten gleich viele Stimmen bekommen haben, viele Wählerkombinationen ausgeschlossen werden, wodurch die Vote Privacy beeinträchtigt wird.

Aus diesen Gründen wurde das Tie-Breaking in Ordinos so realisiert, dass bereits vor der Stimmauszählung für jeden möglichen Gleichstand festgelegt wird, wie dieser bei Bedarf aufgelöst wird. Bei manchen Strategien wie dem Alter der Kandidaten ist dies schon vor der Auswertung bekannt, bei anderen wie der Entscheidung durch externe Personen verlangt Ordinos, dass es vor der Abstimmung festgelegt wird. Der Fall, dass Neuwahlen stattfinden müssen, kann nicht abgedeckt werden, aber durch ein System mit mehreren Runden wie Instant-Runoff [25] simuliert werden. Das Auflösen durch Zufall wird noch extra behandelt.

Das Tie-Breaking wird realisiert, indem für jede Stelle, an der theoretisch ein Gleichstand vorkommen kann, vor der Wahl ein Ranking aller Kandidaten eingetragen wird, das angibt, wie die Kandidaten untereinander bei einem Gleichstand zu gewichten sind. Das Ranking ist eine Permutation der Zahlen $0, ..., n_{cand}$, wobei bei einem Gleichstand der Kandidat mit dem höheren Rank ausgewählt wird. Um die Sicherheitseigenschaften zu gewährleisten, darf das Ranking nur verschlüsselt vorliegen.

Auch bei einer Entscheidung durch Zufall ist es wichtig, dass das Ranking nicht bekannt ist, sondern nur verschlüsselt auf dem Bulletin Board steht. Hierfür gibt es je nach Einstellung von Ordinos zwei Möglichkeiten: Wenn in der Setup-Phase bereits eine ehrliche Partei existiert, die beispielsweise auch für die Schlüsselgenerierung zuständig ist [31], kann diese die zufälligen Permutationen erzeugen, verschlüsseln und auf das Bulletin Board legen. Wegen der Ehrlichkeit ist garantiert, dass die Permutationen zufällig sind und nur die ehrliche Partei sie kennt. Diese Variante wurde im Zuge der Bachelorarbeit implementiert.

Ordinos kann auch ohne diese ehrliche Partei aufgesetzt werden und die Schlüsselgenerierung findet als verteiltes Protokoll unter den Trustees statt [31]. In diesem Fall müssen die Trustees auch die zufälligen geheimen Permutationen berechnen. Dafür gibt es in der Literatur Standardverfahren, wie beispielsweise Mix-Nets, die verwendet werden können [43]. Da in dieser Arbeit der Schwerpunkt auf der eigentlichen Auswertung der Wahl liegt, wurde dieses Verfahren nicht implementiert, aber es ist trotzdem mit dem Tie-Breaking im folgenden Kapitel kompatibel.

Bei Zufall ist es außerdem wichtig, dass für jede Stelle, an der ein Gleichstand auftreten könnte, eine eigene Permutation erzeugt wird, und nicht nur für jede auftretende Kombination von Kandidaten oder Parteien. Sonst kann es beispielsweise bei mehreren Wahlkreisen, in denen die gleichen Parteien antreten, dazu kommen, dass einzelne Parteien systematisch bevorzugt oder benachteiligt werden.

Diese Strategie kann jedoch im Allgemeinen nur bei Personenwahlen eingesetzt werden. Bei Parlamentswahlen kann es vorkommen, dass ein Gleichstand zwischen verschiedenen Parteien aufgelöst werden muss, der in unterschiedlichen Sitzverteilungen resultiert, ohne dass zu diesem Zeitpunkt bekannt ist, wer ein Mandat für diesen Sitz bekommt. Ein Beispiel hierfür ist die Bundestagswahl. In diesem Fall kann das Ranking abhängig von Eigenschaften der Parteien sein, indem beispielsweise die Partei mit mehr Stimmen bevorzugt wird [4] oder es kann durch Zufall entschieden werden [46].

Für Ordinos wurde in der Setup-Phase hinzugefügt, dass die ehrliche Partei alle benötigten zufälligen Permutationen vorberechnet. Dafür muss vor jeder Wahl eine obere Schranke angegeben werden, wie viele Permutationen maximal für jede möglicherweise vorkommende Anzahl an Parteien beziehungsweise Kandidaten benötigt wird. Da diese Berechnungen in der Setup-Phase vor der eigentlichen Auswertung geschehen, ist es unproblematisch, wenn zu viele Permutationen vorberechnet werden, da die Laufzeit der Setup-Phase nicht in die finale Laufzeit der Auswertung einfließt. Anschließend stellt die ehrliche Partei einen Iterator bereit, der den Trustees bei der Auswertung jeweils neue Permutationen für die benötigte Anzahl an Kandidaten zurückgibt, damit die Trustees mit diesen den Gleichstand auflösen können.

5.4.2 Durchführung des Tie-Breakings

Um einen Gleichstand aufzulösen, werden die Stimmen der Kandidaten, zwischen denen Gleichstand herrschen könnte, mit den Tie-Breaking-Ranks in Punkte umgerechnet. Für das Tie-Breaking werden zwei Möglichkeiten in Betracht gezogen. Einerseits kann ähnlich wie in Quelle [10] vor der eigentlichen Auswertung verhindert werden, dass Kandidaten die gleiche Punktzahl haben. Dazu werden die aggregierten Stimmzahlen v_i jedes Kandidaten folgendermaßen in Punkte umgewandelt:

$$p_i = v_i \cdot n_{cand} + r_i \tag{5.1}$$

Hier bezeichnen n_{cand} die Anzahl an Kandidaten und r_i den Rank des i-ten Kandidaten. Da alle Ränge unterschiedlich sind, bewirkt die Umformung, dass keine Kandidaten die gleiche Anzahl an Punkten haben, aber ein Kandidat mit einer höheren Stimmzahl auch eine höhere Punktzahl erhält. Nach dieser Umformung können die Punkte anstatt der Stimmen an die eigentliche Auswertungsfunktion, die beispielsweise den Kandidaten mit den meisten Stimmen bestimmt, übergeben werden. Der Vorteil dieser Methode ist, dass abgesehen von der Berechnung der Punkte keine weiteren, möglicherweise aufwändigen, Berechnungen durchgeführt werden müssen, um den Gleichstand aufzulösen. Ein Nachteil ist aber, dass die Bit-Größe der Punkte um den Faktor n_{cand} größer als die der Stimmen sind, was vor allem bei vielen Kandidaten dazu führen kann, dass Vergleiche mehr Laufzeit benötigen.

Die andere Methode ist, zuerst den Gleichstand zu erkennen und diesen dann gezielt aufzulösen. Dabei wird zuerst eine modifizierte Auswertung der Stimmzahlen vorgenommen, bei der alle möglichen Gewinner bestimmt werden. Als Ergebnis wird pro Kandidat ein Wert t_i zurückgegeben, der "eins" ist, wenn der Kandidat zu den vorläufigen Gewinnern gehört, zwischen denen Gleichstand herrscht und "null", wenn nicht. Aus den Gewinnern, zwischen denen Gleichstand herrscht, wird anschließend ein eindeutiger Gewinner durch folgende Umformung gebildet:

$$p_i = t_i \cdot (r_i + 1) \tag{5.2}$$

Aus diesen Punkten muss anschließend das Maximum bestimmt werden und der zugehörige Kandidat ist der Gewinner der Auswertung. Diese Methode hat den Vorteil, dass die verschlüsselten Klartexte im Vergleich zur vorherigen Methode deutlich kleiner bleiben. Insbesondere bei der Bestimmung des Maximums der Punkte sind die Werte durch die Anzahl der Kandidaten begrenzt, weshalb die Vergleiche eher günstig sind. Aber es muss zuerst eine modifizierte Auswertung vorgenommen werden, die je nach Verfahren zu längeren Laufzeiten führen kann. Daher muss in der Praxis abhängig von dem Auswertungsverfahren und der Anzahl der Kandidaten beziehungsweise Parteien und Stimmen getestet werden, welche der Methoden zu besseren Laufzeiten führt.

Nun wird gezeigt, dass beim Hinzufügen des beschriebenen Verfahrens für das Tie-Breaking zu Ordinos alle Sicherheitseigenschaften weiterhin gewährleistet sind. Für den Beweis wird von dem implementierten Fall ausgegangen, dass eine ehrliche Instanz die Ranks für das Tie-Breaking erstellt. Für die Verwendung von Ordinos ohne eine ehrliche Instanz muss eine Implementierung für die Generierung der Ranks verwendet werden, die die Sicherheitseigenschaften von Ordinos ebenfalls erfüllt.

Theorem 5 (Sicherheit des Tie-Breakings mit Ordinos)
Sowohl die Generierung der Tie-Breaking-Ranks durch die ehrliche Instanz als auch die Verwendung der berechneten Ranks für das Tie-Breaking im Evaluierungsprotokoll erfüllen die Eigenschaften Tally Hiding, Vote Privacy und Accountability.

Beweis Wie in Theorem 4 begründet, sind die Voraussetzungen (a)-(d) aus Theorem 1 und (a) aus Theorem 2 trivialerweise erfüllt. Somit müssen nur Voraussetzung (e) aus Theorem 1 und Voraussetzung (b) aus Theorem 2 gezeigt werden.

Die Generierung der Ranks für das Tie-Breaking wird in der vorgestellten Implementierung durch eine ehrliche Partei ausgeführt und verschlüsselt auf das Bulletin Board geschrieben. Voraussetzung (e) für das Tally Hiding ist gewährleistet, da die Ranks nur verschlüsselt auf dem Bulletin Board stehen und daher keine zusätzlichen Informationen preisgegeben werden. Dass es sich um eine ehrliche Instanz handelt, garantiert, dass die unverschlüsselten Ranks nicht veröffentlicht werden. Auch bei der Verwendung der Tie-Breaking-Ranks im Evaluierungsprotokoll muss das Tally Hiding erfüllt sein, das heißt die Ranks dürfen auch dort nicht entschlüsselt werden. Dies wird in Kapitel 6 bei den jeweiligen Umsetzungen der Wahlverfahren gezeigt. Dabei werden die Tie-Breaking-Ranks als zusätzliche Eingabewerte behandelt, die ebenfalls nicht entschlüsselt werden dürfen.

Auch Voraussetzung (b) für die Accountability ist erfüllt. Für die ehrliche Instanz entfällt die Accountability, da sie sich nach Definition immer korrekt verhält, und daher keine Fehler oder Manipulationen macht, für die sie verantwortet werden müsste. Trustees, die die Ranks verwenden müssen, können auch accountable gehalten werden, denn die Ranks, die sie als Input verwenden müssen, stehen öffentlich auf dem Bulletin Board und die Trustees veröffentlichen alle verschlüsselten Zwischenergebnisse auf dem Bulletin Board. Da für die konkreten Evaluierungsprotokolle gezeigt wird, dass sie Accountability erfüllen, gilt diese Eigenschaft auch in Bezug auf die Tie-Breaking-Ranks als zusätzlichen Input. ■

5.5 Weitere Unterschiede

Einige der vorgestellten Wahlverfahren aus der Praxis setzen sich aus mehreren Evaluierungsfunktionen zusammen. Die Bundestagswahl nutzt beispielsweise als Zwischenschritte mehrere Sainte-Laguë-Evaluierungen. Da solche Zwischenergebnisse wegen des Tally Hidings nicht entschlüsselt werden dürfen, müssen für die Evaluierungsfunktionen Varianten implementiert werden, die das Ergebnis nicht entschlüsseln.

Die Laufzeit ist bei politischen Wahlverfahren mit realen Eingabewerten im Allgemeinen höher, da viele Kandidaten bzw. Parteien antreten und viele Wähler ihre Stimmen abgeben. Bei einigen der implementierten Wahlverfahren ist die Laufzeit merklich abhängig von der Anzahl der Kandidaten und beispielsweise in Quelle [25] wurde bei einigen Verfahren nur bis zu fünf Kandidaten gebenchmarkt, während bei realen Wahlen über 20 antretende Parteien möglich sind [46], [27]. Daher und weil beispielsweise bei der deutschen Bundestagswahl und der Wahl des norwegischen Parlaments mehrere Iterationen von Sitzzuteilungsverfahren vorkommen, ist mit eher längeren Laufzeiten zu rechnen. Für diese Arbeit wurde sich dazu entschieden, dass Laufzeiten bis etwa zwei Tagen noch akzeptabel sind, da die Verfahren komplex sein können, und zwei Tage als Wartezeit auf das Ergebnis einer Wahl als akzeptabel erscheint, wenn beachtet wird, dass die Auswertung von Ordinos die Vorteile des Tally Hidings, der Vote Privacy und der Accountability bringt.

Bei der Umsetzung von Wahlverfahren aus der Praxis können auch die konkreten Gegebenheiten der Wahl ausgenutzt werden, da die Verfahren nicht auf andere Wahlen erweiterbar sein müssen. So können beispielsweise Eigenschaften des Sainte-Laguë-Verfahren genutzt werden, die abhängig von der Mindestsitzzahl pro Partei sind. Da beispielsweise bei der Wahl des norwegischen Parlaments die 4%-Hürde gilt und 169 Sitze verteilt werden, ist die Mindestsitzzahl von zwei Sitzen pro Partei immer erfüllt, doch bei anderen Wahlen ohne Sperrklausel kann nicht davon ausgegangen werden.

6 Umsetzung der ausgewählten Verfahren mit Ordinos

Dieses Kapitel beschreibt die Umsetzung mit Ordinos der Wahlverfahren für das britische House of Commons, den Deutschen Bundestag sowie das norwegische Parlament. Zusätzlich werden verschiedene Varianten des Sainte-Laguë-Verfahrens umgesetzt, die als Grundlage für die Wahlverfahren des deutschen und des norwegischen Parlaments dienen.

Da in Ordinos wegen des Tally Hidings die Auswertung in verschlüsselter Form erfolgt, müssen die Algorithmen dementsprechend optimiert werden, wie es bereits in Kapitel 2.4.1 beschrieben wurde. Daher können die Verfahren in den meisten Schritten nicht genau so implementiert werden, wie sie in Kapitel 4 definiert sind, sondern es müssen eigene Verfahren entwickelt werden, die zum korrekten Ergebnis kommen, aber für Ordinos optimiert sind. Dies wird beispielsweise bei der Umsetzung des Sainte-Laguë-Verfahrens gezeigt. Hier wird neben den für den weiteren Verlauf verwendeten Varianten auch ein naiver Ansatz untersucht, der sich stark an die ursprüngliche Definition des Höchstzahlverfahrens aus Kapitel 4.8.1 anlehnt. Trotz einiger Optimierungen ist die Laufzeit dieses Ansatzes bereits für kleine Input-Größen deutlich zu hoch, weshalb neue Ansätze für das Höchstzahlverfahren entwickelt werden müssen, die auch bei den Anforderungen von Ordinos zu akzeptablen Laufzeiten führen.

Zusätzlich wird in diesem Kapitel gezeigt, dass alle Evaluierungsfunktionen die Eigenschaften Tally Hiding, Vote Privacy und Accountability erfüllen und wenn nötig, dass die Verfahren zum richtigen Ergebnis führen. Abschließend wird durch Benchmarks bestätigt, dass die Verfahren so effizient sind, dass sie auch bei realen Wähler- und Kandidatengrößen für die politischen Wahlen eingesetzt werden können. Bei den Benchmarks wird von der Parallelisierung ausgegangen, die im Verfahren beschrieben wurde. Im Allgemeinen wird nur dann parallelisiert, wenn dieselbe Berechnung für mehrere Wahlkreise oder Parteien ausgeführt werden muss. In der Ordinos-Software sind die Parallelisierungen nicht implementiert, da die Trustees und die arithmetisch-logischen Grundprotokolle noch keine Parallelisierung unterstützen, und dies den Rahmen der Bachelorarbeit überschreiten würde. Daher werden bei den Benchmarks die Algorithmen sequenziell getestet und die Parallelisierung nur rechnerisch ermittelt.

6.1 Umsetzung des Sitzzuteilungsverfahrens nach Sainte-Laguë

Das Sainte-Laguë-Verfahren ist zwar keine eigene Wahl, wird hier aber dennoch ausführlich behandelt, da es ein eigenständiges Verfahren ist und in mehreren implementierten Wahlen verwendet wird. Insbesondere wird es sowohl bei der Bundestagswahl als auch bei der norwegischen Wahl mehrmals innerhalb der Auswertung angewandt. Allein bei der offiziellen Auswertung der letzten

Bundestagswahl wird das Sainte-Laguë-Verfahren 23-mal verwendet [46]. Daher ist es in diesem Abschnitt besonders wichtig, die Laufzeit bestmöglich zu optimieren, denn durch das häufige Anwenden innerhalb der Auswertung für eine Wahl wird jede Laufzeitverbesserung vervielfacht.

Sowohl das Höchstzahlverfahren als auch das Divisorverfahren können nicht direkt mit Tally Hiding in Ordinos implementiert werden. Einerseits haben beide Verfahren das Problem, dass mit rationalen Zahlen gerechnet wird, während Ordinos durch die Paillier-Verschlüsselung nur natürliche Zahlen darstellen kann. Insbesondere für das Tie-Breaking ist nichtsdestotrotz relevant, ob zwei rationale Zahlen exakt gleich sind oder ob eine größer ist, weshalb auch Runden keine Option darstellt.

Zusätzlich ist der Divisor für das Divisorverfahren im Allgemeinen nicht bekannt und eine iterative Berechnung geht zu Lasten der Laufzeit, weil keine kleine Obergrenze für die Anzahl an Iterationen gefunden wurde. Ein weiterer Nachteil des Divisorverfahrens ist, dass wie in Abschnitt 4.8.2 beschrieben bei Gleichstand nicht unabhängig durch Zufall aufgelöst werden kann.

Daher wird im allgemeinen Fall das Höchstzahlverfahren umgesetzt. Dazu werden ein naiver Ansatz, ein verbessertes Basis-Höchstzahlverfahren und ein Verfahren für große Sitzzahlen und wenig Parteien entwickelt. Zusätzlich wird das modifizierte Höchstzahlverfahren für die Wahl des norwegischen Parlaments umgesetzt. Da keines der Verfahren für alle Anwendungsfälle geeignet ist, ist es nötig, alle unterschiedlichen Varianten zu nutzen, sodass für jedes Auftreten des Verfahrens in der Auswertung einer Wahl die jeweils beste Variante verwendet werden kann. Auch wenn das Sainte-Laguë-Verfahren alleinstehend keines der beschriebenen Wahlverfahren aus der Praxis ist, werden die Eigenschaften von großen politischen Wahlverfahren aus Praxis trotzdem angewandt, weshalb abgesehen von dem naiven Ansatz alle Varianten das Tie-Breaking umsetzen und auf große Anzahlen von Wähler, Kandidaten und Sitzen gebenchmarkt wurden. Dies dient dazu, dass das Sainte-Laguë-Verfahren ohne weitere Änderungen von den Evaluierungsfunktion der Bundestagswahl und der Wahl des norwegischen Parlaments aufgerufen werden kann.

6.1.1 Naiver Ansatz

Der naive Ansatz implementiert das Höchstzahlverfahren vergleichsweise direkt. Es wird die in Abschnitt 4.8.1 beschriebene Tabelle aufgestellt und die n_{seats} größten Werte bestimmt. Anschließend wird für jede der n_{seats} größten Höchstzahlen der zugehörigen Partei ein Sitz zugeteilt.

Darstellung der Höchstzahlen

Das Problem der Höchstzahlen ist, dass es sich um rationale Zahlen handelt. Um nur mit natürlichen Zahlen zu rechnen, wurden zwei Lösungswege in Betracht gezogen. Im ersten Fall werden alle Zahlen mit einem Ausgleichsfaktor multipliziert, sodass garantiert ist, dass alle Produkte natürliche Zahlen sind. Der zweite Weg ist, direkt mit Brüchen zu rechnen und Zähler und Nenner getrennt zu betrachten.

Für die erste Variante werden zuerst alle Höchstzahlen modifiziert, indem sie mit einem Ausgleichsfaktor multipliziert werden. Seien v_i die Anzahl der Stimmen für Partei i, $h_{i,k} = \frac{v_i}{2 \cdot k - 1}$ die im Verfahren beschriebenen Höchstzahlen und $\tilde{h}_{i,k}$ die zugehörige modifizierte Höchstzahl.

Der Ausgleichsfaktor ist so definiert:

$$a = \prod_{l=1}^{n_{seats}} (2 \cdot l - 1) \tag{6.1}$$

Daraus können die modifizierten Höchstzahlen berechnet werden:

$$\tilde{h}_{i,k} = a \cdot h_{i,k} \tag{6.2}$$

Da nach 6.2 alle Höchstzahlen durch Multiplikation mit dem gleichen Faktor modifiziert werden, ändert sich die größer-gleich-Beziehung der Höchstzahlen untereinander nicht und das Höchstzahlverfahren kann wie in Abschnitt 4.8.1 beschrieben angewandt werden. Der Nachteil der Ausgleichsfaktoren ist aber, dass die Bit-Größe der verschlüsselten Klartexte groß wird und bei realen Wähler- und Parteigrößen bei Bit-Längen von über 1000 erreicht, wodurch Vergleichsoperationen sehr teuer werden.

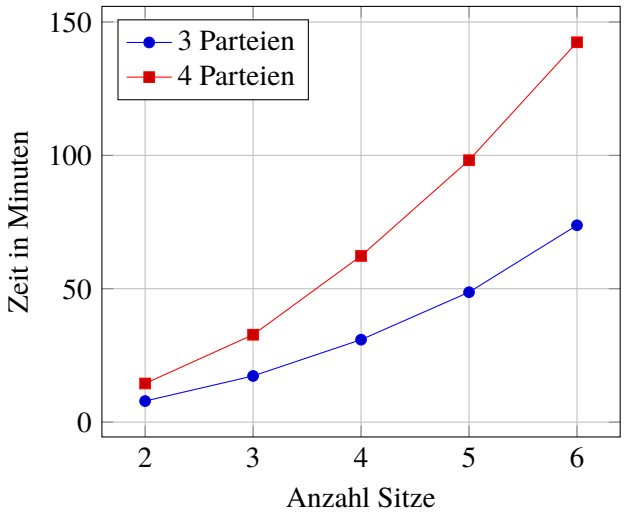
Die zweite Variante löst dieses Problem, indem jeder Bruch als Zähler und Nenner gespeichert wird. Um mit Brüchen zu rechnen, wurde eine arithmetische Black Box für Brüche geschrieben, die arithmetische Grundoperationen und Vergleiche enthält und auf den Grundoperationen für Chiffretexte basiert. Dies hat den Vorteil, dass beispielsweise bei einem Vergleich nicht mehr zwischen allen Brüchen der Tabelle ausgeglichen werden muss, sondern nur mit den Nennern der beiden zu vergleichenden Brüche multipliziert wird, wodurch die maximale Bitlänge deutlich reduziert wird. Da die Benchmarks gezeigt haben, dass die Variante mit Brüchen schneller ist, wurde sich für diese entschieden. Die Black Box wird auch für weitere Implementierungen verwendet, in denen mit Brüchen gerechnet wird, aber dort nicht mehr explizit erwähnt.

Optimiertes Sortieren

Der nächste Schritt des Höchstzahlverfahrens ist, die Höchstzahlen zu sortieren, um die n_{seats} größten Höchstzahlen zu erreichen. In [25] wurde bereits ein Algorithmus beschrieben, der die n_{seats} größten Elemente einer Liste zurückgibt und hier verwendet wird. Für das Höchstzahlverfahren hat die Liste $n_{parties} \cdot n_{seats}$ Elemente, da die Tabelle der Höchstzahlen $n_{parties}$ Spalten hat und n_{seats} Zeilen haben muss, da im Worst Case eine Partei alle Sitze bekommt. Dann werden alle Elemente der Liste paarweise verglichen und für jedes Element wird gespeichert, wie oft es größer als ein anderes Element ist. Dadurch enthält man eine Permutation der Werte $\mathrm{E_{pk}}(0), \mathrm{E_{pk}}(1), \ldots, \mathrm{E_{pk}}(n_{parties} \cdot n_{seats} - 1)$

Allerdings muss nicht jedes Element der Tabelle mit jedem anderen verglichen werden, sondern die Anzahl der Vergleiche kann an zwei Stellen reduziert werden. Zunächst wird eine Spalte der Höchstzahlen betrachtet: Dabei fällt auf, dass die Zahlen nach unten hin immer kleiner werden. Nach Konstruktion gilt dies immer für zwei Elemente der gleichen Spalte, da die Zähler gleich bleiben, aber die Nenner nach unten hin immer größer werden.

Die Vergleiche lassen sich weiter reduzieren, indem die Zeilen betrachtet werden. In jeder Zeile ist der Nenner gleich, aber der Zähler ist abhängig von der Partei. Dabei ist der Zähler aber zwischen den Zeilen gleich, sodass die Sortierung pro Zeile nur ein mal ausgerechnet werden muss und dann auf alle anderen Zeilen übertragen werden kann.



Parameter	Zeit
3 Parteien, 10 Sitze	3h, 25 min
4 Parteien, 10 Sitze	6h, 44 min
3 Parteien, 20 Sitze	13h, 48 min

Abbildung 6.1: Benchmark der naiven Umsetzung des Höchstzahlverfahrens: Der Algorithmus liefert nur bei kleinen Sitz- und Parteizahlen Ergebnisse innerhalb angemessener Laufzeit.

Naiver Algorithmus

Insgesamt ist der naive Algorithmus wie folgt aufgebaut: Zuerst wird die Sortierung der Spalten berechnet und die Vergleiche innerhalb einer Spalte und innerhalb einer Zeile können ohne Berechnung direkt eingetragen werden. Anschließend werden die Elemente paarweise verglichen, deren Ergebnis noch nicht feststeht und für das jeweils größere Element die Anzahl der gewonnenen Vergleiche inkrementiert. Im nächsten Schritt wird die Anzahl der Sitze pro Partei bestimmt. Dabei bekommt jede Partei pro Höchstzahl, die mindestens $n_{seats} \cdot (n_{parties} - 1)$ ist, einen Sitz. Zurückgegeben wird ein Array mit der verschlüsselten Sitzanzahl pro Partei.

Der Algorithmus lässt sich auch auf den Fall anwenden, dass die Anzahl an zu verteilenden Sitzen nur verschlüsselt vorliegt, wenn eine maximale Anzahl an Sitzen angegeben wird. In diesem Fall hat die Tabelle die maximale Sitzanzahl als Zeilenanzahl.

Die Benchmarks in Abbildung 6.1 zeigen aber, dass der Algorithmus verhältnismäßig hohe Laufzeiten hat. Dies liegt daran, dass die Anzahl an Vergleichen quadratisch in $O(n_{seats} \cdot n_{parties})$ ist. Daher kann der naive Ansatz nur für niedrige Sitz- und Parteienzahlen angewandt werden. Dies ist in den ausgewählten Wahlen der Praxis nicht der Fall, weshalb der naive Ansatz nicht verwendet wird.

6.1.2 Basis-Höchstzahlverfahren

Diese Implementierung des Höchstzahlverfahrens hat zum Ziel, die Laufzeit deutlich zu verbessern, indem nicht wie in der naiven Variante alle Höchstzahlen miteinander verglichen werden, sondern pro Sitz nur so viele Vergleiche benötigt werden, wie es Parteien gibt. Zusätzlich wird für das

	Partei A	Partei B	Partei C
$\frac{v_i}{1}$	100	**150**	90
$\frac{v_i}{3}$	33,3	50	30
$\frac{v_i}{5}$	20	30	18
$\frac{v_i}{7}$	14,3	21,4	12,9

	Partei A	Partei B	Partei C
$\frac{v_i}{1}$	**100**	150	90
$\frac{v_i}{3}$	33,3	**50**	30
$\frac{v_i}{5}$	20	30	18
$\frac{v_i}{7}$	14,3	21,4	12,9

	Partei A	Partei B	Partei C
$\frac{v_i}{1}$	100	150	**90**
$\frac{v_i}{3}$	33,3	50	30
$\frac{v_i}{5}$	20	30	18
$\frac{v_i}{7}$	14,3	21,4	12,9

Abbildung 6.2: Das Beispiel zeigt die ersten drei Iterationsschritte des Basis-Höchstzahlverfahrens. In rot ist die Menge current elements dargestellt, die immer die nächstgrößere Höchstzahl enthält. Diese ist markiert und wird in jedem Schritt ersetzt. Im Beispiel hat nach den drei Iterationen jede Partei Anspruch auf einen Sitz.

Basis-Höchstzahlverfahren auch das Tie-Breaking umgesetzt und gezeigt, dass es die Sicherheitseigenschaft von Ordinos erfüllt und daher in Ordinos als Baustein für Wahlverfahren aus der Praxis verwendet werden kann. Außerdem gibt es zwei Varianten dieses Verfahrens für den Fall, dass die Anzahl an Gesamtsitzen verschlüsselt ist, und für die skandinavischen Variante.

Das Basis-Höchstzahlverfahren funktioniert folgendermaßen: Zuerst wird die Tabelle aus Abschnitt 4.8.1 aufgestellt. Zusätzlich wird eine Menge `current elements` der Tabelleneinträge benötigt, die für die aktuelle Iteration relevant sind, das heißt, unter denen die nächstgrößere Höchstzahl garantiert enthalten ist. Diese Menge enthält immer genauso viele Elemente, wie es Parteien gibt. Initial besteht `current elements` aus den Einträgen der ersten Zeile der Tabelle. Anschließend wird in einer Schleife durch alle zu verteilenden Sitze iteriert. Für jeden Sitz wird das größte Element von `current elements` bestimmt und die zugehörige Partei enthält einen Sitz mehr. Danach wird `current elements` aktualisiert, indem das verwendete größte Element durch den Tabelleneintrag eine Zeile tiefer in der Tabelle ersetzt wird. Die ersten Iterationen dieses Algorithmus werden in Abbildung 6.2 dargestellt.

Die Implementierung folgt diesem Vorgehen vergleichsweise direkt und die Höchstzahlen werden als Brüche mit Zähler und Nenner gespeichert. Die Tabelle wird nicht explizit definiert, sondern es werden nur die aktuellen Elemente gespeichert und die nächstgrößere Höchstzahl wird berechnet, indem der Nenner der aktuellen Höchstzahl um zwei vergrößert wird. Eine Iteration dieses Verfahrens ist in Algorithmus 6.1 als Pseudocode beschrieben. Dort wird die Methode `getMaxFractionAsIndex` genutzt. Diese bestimmt in $O(n)$ den größten Bruch und gibt eine Liste an verschlüsselten Indizes zurück, wobei das größte Element den Index ”eins” und alle anderen den Index ”null” haben.

Hinzufügen des Tie-Breakings

Das Tie-Breaking ist beim Höchstzahlverfahren relevant, wenn wie in Beispiel 4.2 die n_{seats} größten Zahlen nicht eindeutig sind. Der beschriebene Algorithmus 6.1 für eine Iteration des verbesserten Höchstzahlverfahrens setzt kein Tie-Breaking um, da wenn `currentElements` mehrere gleich große Höchstzahlen enthält, das Maximum, das durch `getMaxFractionAsIndex` bestimmt wird, immer das letzte größte Element ist.

Algorithmus 6.1 Iteration des Basis-Höchstzahlverfahrens: Aus der Menge `current elements` wird das größte Element bestimmt und der zugehörigen Partei ein Sitz zugeteilt. Außerdem wird die Menge `current elements` aktualisiert.

procedure AddSeat($E_{pk}(currentElements)$, $E_{pk}(seatsPerParty)$)
  $E_{pk}(indexMax)$ = getMaxFractionAsIndex($E_{pk}(currentElements)$)
  for $p \in parties$ **do** // Update currentElements
    $E_{pk}(el) = E_{pk}(currentElements)[p]$
    $E_{pk}(newDenominator) = E_{pk}(el.denominator) +_e 2 \cdot_e E_{pk}(indexMax)[p]$
    $E_{pk}(currentElements)[p]$ =Fraction($E_{pk}(el.numerator)$, $E_{pk}(newDenominator)$)
  end for
  for $p \in parties$ **do** // Update seats
    $E_{pk}(seatsPerParty)[p] = E_{pk}(seatsPerParty)[p] +_e E_{pk}(indexMax)[p]$
  end for
  return $E_{pk}(currentElements)$, $E_{pk}(seatsPerParty)$
end procedure

Bei den meisten Iterationen beeinträchtigt dies aber das Ergebnis nicht, denn wenn das Maximum nicht uneindeutig ist, wird in einer Iteration eines der Maxima gewählt, der zugehörigen Partei ein Sitz gegeben und dieses Maximum durch ein kleineres Element ersetzt. In der nächsten Iteration ist das andere Element das Maximum und auch diese Partei bekommt einen Sitz. Da das Ergebnis des Höchstzahlverfahrens nur die Summe der Sitze pro Partei ist, ist es nicht von Bedeutung, in welcher Reihenfolge die Sitze vergeben werden.

Daraus folgt, dass eine Uneindeutigkeit des Maximums nur bei der Verteilung der letzten $n_{parties} - 1$ Sitze zu unterschiedlichen Ergebnissen führen kann und nur in diesem Fall das Tie-Breaking angewandt werden muss. Im Worst Case sind gegen Ende der Iterationen alle Elemente der Menge `currentElements` gleich groß. Zwischen mehr Elementen kann kein Gleichstand herrschen, da wenn ein Element als Maximum ausgewählt und ersetzt wird, das neue Element den gleichen Zähler, aber einen größeren Nenner hat. Daher muss das neue Element kleiner sein. (Im Fall, dass der Zähler, also die Anzahl an Stimmen, null ist, kann das ursprüngliche Element nicht das Maximum sein.) Daher können die ersten Iterationen ohne Tie-Breaking durchgeführt werden und nur bei den letzten $n_{parties} - 1$ Iterationen ist Tie-Breaking notwendig.

Wie in Abschnitt 5.4.2 beschrieben, gibt es die Möglichkeit vor der Auswertung den Gleichstand aufzulösen oder erst den Gleichstand zu erkennen und ihn dann gezielt aufzulösen. Für das Höchstzahlverfahren wurden beide Varianten in Betracht gezogen und ihre Laufzeit verglichen.

Die erste Variante für eine Iteration mit Tie-Breaking läuft folgendermaßen ab: Anstatt nur ein Maximum auszuwählen, werden alle Maxima gefunden, indem zusätzlich ein equals-Vergleich ausgeführt wird. Anschließend werden die Ranks für das Tie-Breaking verwendet, wie sie an Abschnitt 5.4.1 definiert werden. Daraus werden für jede Partei Punkte berechnet, die null sind, wenn die Höchstzahl kein Maximum ist und sonst $rank[i] + 1$ entsprechen. Nun wird der Partei mit der höchsten Punktzahl ein Sitz zugeordnet und wegen des Tie-Breakings ist diese Partei eindeutig.

Die zweite Variante wird in Algorithmus 6.2 beschreiben. Hier werden vor der Bestimmung des größten Elements die Brüche in eine eindeutige Reihenfolge gebracht, indem jeder Bruch mit der Anzahl der Elemente in `current elements` multipliziert wird und anschließend die Ranks für das Tie-

Algorithmus 6.2 Iteration des Basis-Höchstzahlverfahrens mit Tie-Breaking: Für das Tie-Breaking werden die Elemente mit Ranks modifiziert, sodass die Reihenfolge zwischen zwei Elementen eindeutig festgelegt wird. Anschließend wird wie bei der Iteration ohne Tie-Breaking vorgegangen.

procedure AddSeatTieBreaking($\mathrm{E}_{\mathrm{pk}}(currentElements)$, $\mathrm{E}_{\mathrm{pk}}(seatsPerParty)$)
  for $p \in parties$ **do**
    $\mathrm{E}_{\mathrm{pk}}(el) = \mathrm{E}_{\mathrm{pk}}(currentElements)[p]$
    $\mathrm{E}_{\mathrm{pk}}(num) = \mathrm{E}_{\mathrm{pk}}(el.numerator) \cdot_e n_{parties} +_e \mathrm{E}_{\mathrm{pk}}(ranks)[p] \cdot_e \mathrm{E}_{\mathrm{pk}}(el.denominator)$
    $\mathrm{E}_{\mathrm{pk}}(modElements)[p] = \mathrm{Fraction}(\mathrm{E}_{\mathrm{pk}}(num), \mathrm{E}_{\mathrm{pk}}(el.denominator))$
  end for
  $\mathrm{E}_{\mathrm{pk}}(indexMax) = \mathrm{getMaxFractionAsIndex}(\mathrm{E}_{\mathrm{pk}}(modElements))$
  for $p \in parties$ **do** // Update currentElements
    $\mathrm{E}_{\mathrm{pk}}(el) = \mathrm{E}_{\mathrm{pk}}(currentElements)[p]$
    $\mathrm{E}_{\mathrm{pk}}(newDenominator) = \mathrm{E}_{\mathrm{pk}}(el.denominator) +_e 2 \cdot_e \mathrm{E}_{\mathrm{pk}}(indexMax)[p]$
    $\mathrm{E}_{\mathrm{pk}}(currentElements)[p] = \mathrm{Fraction}(\mathrm{E}_{\mathrm{pk}}(el.numerator), \mathrm{E}_{\mathrm{pk}}(newDenominator))$
  end for
  for $p \in parties$ **do** // Update seats
    $\mathrm{E}_{\mathrm{pk}}(seatsPerParty)[p] = \mathrm{E}_{\mathrm{pk}}(seatsPerParty)[p] +_e \mathrm{E}_{\mathrm{pk}}(indexMax)[p]$
  end for
  return $\mathrm{E}_{\mathrm{pk}}(currentElements), \mathrm{E}_{\mathrm{pk}}(seatsPerParty)$
end procedure

Anzahl der Parteien	Zeit Variante 1	Zeit Variante 2
2	90s	61s
6	8min, 47s	5min, 12s
20	44 min, 45s	25min, 37s

Tabelle 6.1: Die Laufzeiten der beiden Varianten für eine Iteration mit Tie-Breaking werden für große Werte ($n_{voters} = 1.000.000$) verglichen. Die erste Variante erkennt zuerst die Gleichstände, um sie gezielt aufzulösen, während die zweite Variante bereits vor der Auswertung den Gleichstand auflöst. Es ist deutlich zu erkennen, dass die zweite Variante für alle Parteienanzahlen schneller ist und es ist zu erwarten, dass dies auch für nicht getestete Werte gilt.

Breaking addiert werden. Dadurch bleibt die Reihenfolge von ungleichen Elementen erhalten, aber gleichen Elementen wird durch die Tie-Breaking-Ranks eine eindeutige Reihenfolge zugewiesen. Ein Nachteil dieses Verfahrens ist, dass die Elemente dadurch sehr groß werden können. Dies könnte zu einer längeren Laufzeit des Verfahrens führen, da die benötigte Zeit für Vergleichsoperationen abhängig von der maximalen Bitlänge der zu vergleichenden Werte ist.

Beim Vergleich der beiden Varianten ist zu erwarten, dass für kleine Wähler- und Parteiengrößen die zweite Variante schneller ist, da die Zähler der modifizierten Elemente nicht so groß werden, dass die Vergleichsoperationen merklich länger dauern. Nur bei großen Zahlen kann es vorkommen, dass die erste Variante schneller ist. Um dies zu überprüfen wurde mit beiden Varianten ein Benchmark durchgeführt. Die Ergebnisse in Tabelle 6.1 zeigen deutlich, dass die zweite Variante schneller ist, weshalb diese für das Tie-Breaking des Höchstzahlverfahrens verwendet wird.

Algorithmus 6.3 Basis-Höchstzahlverfahren: In jeder Iteration wird ein Sitz an die durch `addSeat` ausgewählte Partei verteilt und nur bei den letzten Iterationen ist Tie-Breaking nötig.

procedure HÖCHSTZAHLVERFAHREN(n_{seats}, $\mathrm{E}_{\mathrm{pk}}(votes)$)
  for $p \in parties$ **do**
    $\mathrm{E}_{\mathrm{pk}}(currentElement)[p] = \textsc{Fraction}(\mathrm{E}_{\mathrm{pk}}(votes)[p],1)$
    $\mathrm{E}_{\mathrm{pk}}(seats)[p] = \mathrm{E}_{\mathrm{pk}}(0)$
  end for
  for $i = 1, \ldots, n_{seats} - n_{parties} + 1$ **do**
    $\mathrm{E}_{\mathrm{pk}}(currentElements), \mathrm{E}_{\mathrm{pk}}(seats) = \textsc{addSeat}(\mathrm{E}_{\mathrm{pk}}(currentElements), \mathrm{E}_{\mathrm{pk}}(seats))$
  end for
  for $i = n_{seats} - n_{parties} + 2, \ldots, n_{seats}$ **do**
    $\mathrm{E}_{\mathrm{pk}}(currentElements), \mathrm{E}_{\mathrm{pk}}(seats) = \textsc{addSeatTieBreaking}(\mathrm{E}_{\mathrm{pk}}(currentElements), \mathrm{E}_{\mathrm{pk}}(seats))$
  end for
  return $\mathrm{E}_{\mathrm{pk}}(seats)$
end procedure

Das gesamte Verfahren ist in Algorithmus 6.3 beschrieben. Dabei sind die ersten Iterationen ohne Tie-Breaking und nur bei den letzten kritischen Iterationen wird mit Tie-Breaking gearbeitet. Er stellt eine deutliche Verbesserung zum naiven Höchstzahlverfahren dar, da statt $O(n^2_{parties} \cdot n^2_{seats})$ jetzt nur noch $O(n_{seats} \cdot n_{parties})$ Vergleiche benötigt werden. Dies zeigt sich auch in den Benchmarks in Abschnitt 6.1.2

Basis-Höchstzahlverfahren mit verschlüsselter Sitzzahl

Das Basis-Höchstzahlverfahren kann auch mit verschlüsselter Sitzzahl durchgeführt werden, wenn eine Obergrenze für die Anzahl an Sitzen bekannt ist. In diesem Fall wird zusätzlich noch ein Indikator benötigt, der ”eins” ist, wenn noch ein Sitz hinzugefügt werden soll und sonst ”null”. Dieser wird berechnet, indem zuerst linear durch ein mit verschlüsselten Nullen gefülltes Array mit der Länge der Maximalsitzzahl iteriert wird und bei jedem Index ein equals-Vergleich durchgeführt wird, ob der aktuelle Index der verschlüsselten Sitzzahl entspricht. Wenn dies der Fall ist, wird das zugehörige Element auf ”eins” gesetzt. Anschließend wird rückwärts durch das Array iteriert und ab dem Index mit der verschlüsselten Eins werden alle Zahlen auf Eins gesetzt. Dieser Schritt ist nur mit Additionen möglich [34]. Beim Basis-Höchstzahlverfahren wird dann in jeder Iteration die neue Sitzzahl berechnet, aber die Sitze werden nur dann aktualisiert, wenn der Indikator ”eins” ist. Da in dieser Variante aber nicht bekannt ist, wann die letzten $n_{parties} - 1$ Iterationen sind, muss in jeder Iteration das Tie-Breaking ausgeführt werden.

Modifizierte Version des Basis-Höchstzahlverfahrens

Die Wahl des norwegischen Parlaments erfordert die modifizierte Version des Sainte-Laguë-Verfahrens, wie sie in Abschnitt 4.8.3 beschrieben wurde. Dabei werden die Stimmen in der ersten Zeile der Tabelle nicht durch 1, sondern durch $1,4$ dividiert. Auch dieses Verfahren wird mit Ordinos umgesetzt.

Im Wesentlichen wird wie beim Basis-Höchstzahlverfahren, das in den letzten Abschnitten beschrieben wurde, vorgegangen. Das Basis-Höchstzahlverfahren nutzt die Menge `current elements`, um durch die relevanten Elemente der Höchstzahlentabelle durchzugehen. Für das modifizierte Verfahren wird dies um eine zweite Tabelle ergänzt, die die erste Zeile der modifizierten Höchstzahlen enthält, in der die Stimmen pro Partei durch $1,4$ dividiert werden. Zusätzlich werden Pointer verwendet, die zeigen, in welcher der beiden Tabellen das jeweilige Element aus `current elements` sich befindet. Dies bewirkt, dass die Menge `current elements` mit durch $1,4$ geteilten Stimmen pro Partei initialisiert wird und sobald ein Element ersetzt wird, wird für dieses in die Tabelle der normalen Höchstzahlen beginnend mit der Division durch drei gewechselt.

Die weitere Tabelle und die benötigten Pointer erhöhen die Laufzeit kaum, da sie mit Additionen und Multiplikationen umgesetzt werden können. Doch da die Division durch $1,4$ als Multiplikation mit $\frac{5}{7}$ dargestellt werden muss, werden für die Vergleiche unter Umständen mehr Bits benötigt, da die obere Schranke des Zählers verfünffacht wird.

Korrektheit und Sicherheitseigenschaften

Im Folgenden wird gezeigt, dass das Basis-Höchstzahlverfahren zu korrekten Ergebnissen führt und die Sicherheitseigenschaften Tally Hiding, Vote Privacy und Accountability erfüllt sind. Diese Ergebnisse gelten auch für die beiden Varianten, das heißt für das Verfahren mit verschlüsselter Gesamtsitzzahl und für das modifizierte Verfahren, da in beiden Fällen nur kleine Änderungen vorgenommen werden, die das Gesamtverfahren nicht grundlegend verändern. Dass die beiden Varianten korrekt sind, wenn die ursprüngliche Umsetzung des Basis-Höchstzahlverfahrens korrekt ist, ist daher einfach zu zeigen.

Theorem 6 (Korrektheit des Basis-Höchstzahlverfahrens)
Das Basis-Höchstzahlverfahren in Algorithmus 6.3 berechnet immer die korrekte Sitzverteilung nach dem Sainte-Laguë-Verfahren. Dies gilt ebenfalls für die Variante mit verschlüsselter Sitzzahl aus Abschnitt 6.1.2 und die Umsetzung der skandinavischen Variante aus Abschnitt 6.1.2 sowie eine Kombination dieser Varianten.

Beweis Die Menge `current elements` enthält immer das nächstgrößere Element, weshalb insgesamt die n_{seats} größten Elemente ausgewählt werden und den zugehörigen Parteien Sitze zugeteilt werden. Dies entspricht genau der Konstruktion des Sainte-Laguë-Verfahrens nach Abschnitt 4.8.1. Außerdem bewirkt das Tie-Breaking, dass auch bei Gleichstand immer genau n_{seats} Elemente ausgewählt werden. ∎

Theorem 7 (Sicherheit des Basis-Höchstzahlverfahrens mit Ordinos)
Das in Abschnitt 6.1.2 und Algorithmus 6.3 vorgestellte Protokoll für das Basis-Höchstzahlverfahren mit Ordinos erfüllt die Sicherheitseigenschaften Tally Hiding, Vote Privacy und Accountability. Außerdem wird das Ergebnis ebenfalls nicht entschlüsselt, weshalb das Basis-Höchstzahlverfahren ohne weitere Änderungen als Grundbaustein für andere Evaluierungsprotokolle verwendet werden kann. Gleiches gilt auch für die beiden Varianten, das Basis-Höchstzahlverfahren mit verschlüsselter Sitzzahl aus Abschnitt 6.1.2 und das modifizierte Basis-Höchstzahlverfahren aus Abschnitt 6.1.2, und für die Kombination beider Varianten.

Beweis Für die Gewährleistung der Sicherheitseigenschaften müssen alle Voraussetzungen aus Theorem 1 für das Tally Hiding und die Vote Privacy und aus Theorem 2 für die Accountability erfüllt sein. Zunächst werden die Voraussetzungen gezeigt, die unabhängig vom konkreten Evaluierungsprotokoll sind: Das Verschlüsselungsschema und der Signierungsalgorithmus sind korrekt und sicher nach den jeweiligen Security Games und die Protokolle für die Schlüsselgenerierung und die Verschlüsselung sind Nicht-Interaktive Zero-Knowledge-Beweise. Dies ist in Ordinos gegeben [31] und wurde für das Basis-Höchstzahlverfahren nicht verändert, weshalb Voraussetzungen (a) und (b) aus Theorem 1 erfüllt sind. Zusätzlich sind Voraussetzungen (c) und (d) aus Theorem 1 und Voraussetzung (a) aus Theorem 2 gegeben, denn der grundsätzliche Aufbau von Ordinos, dass sich genügend Wähler beteiligen, welche Rechte der Angreifer hat und welche Instanzen ehrlich sein müssen, wurde unverändert gelassen. Zusätzlich wurde in Paper [25] gezeigt, dass die verwendeten arithmetischen Grundbausteine bereits alle Sicherheitseigenschaften erfüllen, die Ordinos verlangt. Für das Basis-Höchstzahlverfahren werden diese ohne Änderung wiederverwendet. In Theorem 5 wurde bewiesen, dass die Generierung und Verwendung der Tie-Breaking-Ranks sicher sind.

Somit fehlen nur noch folgende beiden Voraussetzungen zu zeigen: Voraussetzung (e) aus Theorem 1 besagt, dass außer dem Ergebnis keine weiteren Informationen entschlüsselt werden dürfen. Da das aktuelle Theorem aussagt, dass das Basis-Höchstzahlverfahren als Teilprotokoll in anderen Evaluierungsprotokollen verwendet werden darf, muss dies dazu verschärft werden, dass keine Informationen einschließlich des Ergebnisses, entschlüsselt werden dürfen. Dies gilt, da in dem vorgestellten Protokoll keine Eingabewerte, Zwischenergebnisse oder Endergebnisse entschlüsselt werden. Die einzige Ausnahme hiervon sind die Entschlüsselungen der maskierten Werte in den arithmetischen und logischen Grundprotokollen, doch für diese wurde bereits gezeigt, dass die Protokolle sicher sind und dass aus der Entschlüsselung der maskierten Werte keine Informationen abgeleitet werden können [31]. Somit sind das Tally Hiding und die Vote Privacy erfüllt.

Außerdem muss Voraussetzung (b) aus Theorem 2 gezeigt werden, die verlangt, dass überprüft werden kann, wenn der Output nicht das Ergebnis des gegebenen Inputs ist. Dies ist gegeben, da alle Zwischenergebnisse auf das Bulletin Board geschrieben werden und dort von jedem, auch von externen Wahlbeobachtern, einsehbar sind. Dadurch ist sichergestellt, dass die berechneten Ergebnisse eines Grundbausteins korrekt als Eingabewerte des nächsten Grundbausteins verwendet werden. Sollte ein Trustee einen anderen Wert als Eingabe verwenden, wird das durch Nachrechnen erkannt. Damit ist auch die Eigenschaft Accountability erfüllt. ■

Benchmarking

Das Basis-Höchstzahlverfahren ist deutlich besser als der naive Ansatz. Insbesondere ist es mit dem Basis-Höchstzahlverfahren möglich, Sitzverteilungen mit Gesamtsitz- und Parteizahlen zu berechnen, die in der Praxis vorkommen können. In Abbildung 6.3 sind die Benchmarks für bis zu 8 Parteien und bis zu 100 Sitzen dargestellt.

Doch das Basis-Höchstzahlverfahren benötigt für große Anzahlen von Parteien und Sitzen noch immer längere Laufzeiten. Da es bei der Bundestagswahl und der Wahl des norwegischen Parlaments mehrfach verwendet wird, wäre es wünschenswert, die Laufzeit weiter zu reduzieren. Dies trifft vor allem auf den Fall zu, dass viele Sitze verteilt werden müssen, oder dass die Anzahl an Sitzen verschlüsselt ist, denn dann müssen so viele Iterationen ausgeführt werden, wie die Obergrenze ist, obwohl in der Praxis häufig nur deutlich weniger Sitze vergeben werden. Im folgenden Kapitel wird

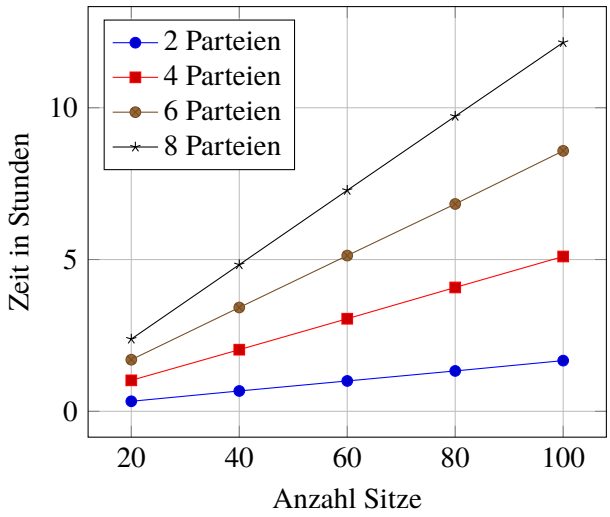


Abbildung 6.3: Benchmark für das Basis-Höchstzahlverfahren: Dieses Verfahren ist deutlich schneller als die naive Implementierung des Höchstzahlverfahrens und kann auch für größere Sitzzahlen gut eingesetzt werden. Die Laufzeit ist linear zu der Anzahl der Sitze, da eine Iteration, um einen Sitz hinzuzufügen, ohne Tie-Breaking konstante Zeit benötigt (abhängig von der Anzahl der Parteien).

daher eine weitere Umsetzung des Höchstzahlverfahrens in Ordinos entwickelt, die auf diesen Fall spezialisiert ist und sich dabei zunutze macht, dass bei Parlamentswahlen die Sitze oft nur zwischen wenigen Parteien verteilt werden.

6.1.3 Höchstzahlverfahren basierend auf der Grundverteilung

Im Paper [25] wird ein anderes Sitzzuteilungsverfahren, das Hare-Niemeyer-Verfahren, vorgestellt, und die Laufzeiten sind hier auch für große Sitzzahlen noch im Minutenbereich [25] und daher deutlich geringer als bei dem Basis-Höchstzahlverfahren. Das Sainte-Laguë-Verfahren und das Hare-Niemeyer-Verfahren funktionieren unterschiedlich, da sie unterschiedliche Parameter optimieren: Das Sainte-Laguë-Verfahren minimiert den quadratischen Fehler der vergebenen Sitze zum exakten Stimmanteil [22], während das Hare-Niemeyer-Verfahren jeder Partei $\left\lfloor \frac{v_i \cdot n_{seats}}{v_{total}} \right\rfloor$ Sitze garantiert [22]. Trotzdem können intuitiv beide Verfahren nicht allzu unterschiedliche Ergebnisse liefern und gleichzeitig den Anspruch haben, die Sitze gerecht basierend auf der Stimmverteilung zu verteilen, da sonst mindestens eines der Verfahren von der Bevölkerung als ungerecht wahrgenommen werden müsste. Auf diesen Überlegungen wird ein Verfahren konstruiert, das eine Sainte-Laguë-Sitzverteilung als Ergebnis liefert, aber für einige Sitze den Mindestsitzanspruch des schnellen Hare-Niemeyer-Verfahrens nutzt.

Algorithmus 6.4 Abgerundete Division: Das Ergebnis ist $\left\lfloor \frac{a}{b} \right\rfloor$ und durch n nach oben beschränkt. Es wird wie bei der binären Suche vorgegangen, um den größten Index zu finden, für den die Bedingung erfüllt ist.

procedure FLOORDIVISION($\mathrm{E_{pk}}(a)$, $\mathrm{E_{pk}}(b)$, n)
  $length$ = BITLENGTH(n)
  $\mathrm{E_{pk}}(lowerIndex) = \mathrm{E_{pk}}(0)$
  for $i = 1, \ldots, length$ **do**
    $\mathrm{E_{pk}}(j) = \mathrm{E_{pk}}(lowerIndex) +_e 2^{length-i}$
    $\mathrm{E_{pk}}(gt) = \mathrm{gt}(a, \mathrm{E_{pk}}(j) \cdot_e \mathrm{E_{pk}}(b))$
    $\mathrm{E_{pk}}(lowerIndex) = \mathrm{E_{pk}}(lowerIndex) +_e 2^{length-i}$
  end for
  return $\mathrm{E_{pk}}(lowerIndex)$
end procedure

Beschreibung des Verfahrens

Zuerst wird eine vorläufige Sitzverteilung berechnet, nach der jede Partei genau $\left\lfloor \frac{v_i \cdot n_{seats}}{v_{total}} \right\rfloor$ Sitze erhält. Dies ist keine Sitzverteilung im eigentlichen Sinne, da die Summe der so zugeordneten Sitze im Allgemeinen kleiner ist als die Gesamtanzahl an Sitzen n_{seats}.

Der Teilalgorithmus für die abgerundete Division $\left\lfloor \frac{a}{b} \right\rfloor$ zweier verschlüsselter Zahlen ist ähnlich wie der in Paper [25] beschriebene Algorithmus, wurde allerdings so optimiert, dass die abgerundete Division in $O(\log(n))$ stattfinden kann. Es wird die größte natürliche Zahl i gesucht, sodass $i \cdot b \leq a$ und für i wird als Obergrenze ein Wert n angegeben. Dabei wird wie bei binärer Suche vorgegangen, um aus den Zahlen zwischen 0 und n die größte Zahl zu finden, die diese Bedingung erfüllt. Das Verfahren ist als Pseudocode in Algorithmus 6.4 beschrieben. Die Idee, wie bei binärer Suche eine Zahl zu finden, die eine bestimmte Bedingung erfüllt, wird auch im nächsten Wahlverfahren bei der Berechnung des Mittelwertes wiederverwendet.

Nachdem für jede Partei der Wert $m_i = \left\lfloor \frac{v_i \cdot n_{seats}}{v_{total}} \right\rfloor$ berechnet wurde, werden daraus zwei Sitzverteilungen bestimmt, wobei mindestens eine der Verteilungen korrekt ist.

Die erste Verteilung ordnet jeder Partei m_i Sitze zu und fügt davon ausgehend weitere Sitze hinzu, bis die Gesamtanzahl an Sitzen im Parlament erhalten ist. Dafür wird das Basis-Höchstzahlverfahren verwendet, aber es wird nicht mit null Sitzen pro Partei, sondern mit m_i Sitzen pro Partei initialisiert. Nun werden wie beim Basis-Höchstzahlverfahren mit verschlüsselter Sitzzahl maximal $n_{parties}$ weitere Sitze schrittweise hinzugefügt.

Anschließend wird getestet, ob die erste Verteilung korrekt ist, das heißt, ob das Ergebnis einer Sitzverteilung nach dem Sainte-Laguë-Verfahren entspricht. Dafür wird die nächste Höchstzahl ermittelt, die aus der Menge `current elements` entnommen werden würde, wenn ein zusätzlicher Sitz benötigt wäre. Dieses Element wird mit den Höchstzahlen der Menge `current elements` zum Zeitpunkt der Initialisierung der ersten Verteilung verglichen und das Minimum bestimmt. Wenn das Minimum die nächste Höchstzahl ist, dann ist die erste Verteilung korrekt, ansonsten ist die zweite Verteilung korrekt.

Bei der zweiten Verteilung werden jeder Partei $m_i + 1$ Sitze zugeordnet, und schrittweise Sitze entfernt, bis die Anzahl an Sitzen des Parlaments erreicht ist. Dabei wird wieder das Basis-Höchstzahlverfahren angewandt, wobei initial jede Partei $m_i + 1$ Sitze statt null Sitze erhält. Bei den einzelnen Iterationen wird nicht wie beim Basis-Höchstzahlverfahren ein Sitz hinzugefügt, sondern ein Sitz entfernt. Das bedeutet, dass immer die kleinste Höchstzahl der Menge `current elements` gefunden wird und durch die Höchstzahl eine Zeile weiter oben in der Tabelle ersetzt wird. Anschließend wird ein Sitz der zugehörigen Partei entfernt. Auch hier ist die Obergrenze der zu entfernenden Sitze $n_{parties}$ und der Ablauf ist analog zum Basis-Höchstzahlverfahren mit verschlüsselter Sitzzahl.

Hier ist zu beachten, dass es nicht möglich ist, einen Sitz von einer Partei zu streichen, die aktuell null Sitze hat. Dies wird so umgesetzt, dass der Nenner der entsprechenden Höchstzahl auf null gesetzt wird. Der Vergleich der arithmetischen Black Box für Brüche bewirkt dann, dass die Höchstzahl nie das Minimum der Menge `current elements` sein kann. Allerdings ist dafür pro Iteration ein weiterer Vergleich nötig, weshalb die Berechnung der zweiten Verteilung mehr Zeit benötigt als die der ersten Verteilung.

Nun sind beide Verteilungen berechnet und es steht fest, welche Verteilung korrekt ist, sodass diese als Ergebnis zurückgegeben werden kann. Der gesamte Ablauf des Höchstzahlverfahrens basierend auf der Grundverteilung ist in Abbildung 6.4 schematisch dargestellt. Bei jedem Hinzufügen und Entfernen eines Sitzes und bei dem Vergleich, welche Verteilung korrekt ist, wird das Tie-Breaking ebenfalls umgesetzt.

Korrektheit des Verfahrens

Da diese Implementierung des Verfahrens deutlich anders als die ursprüngliche Definition in Abschnitt 4.8.1 ist, muss gezeigt werden, dass beide Algorithmen immer die gleiche Sitzverteilung berechnen. Damit das Verfahren auf reale Wahlen angewandt werden kann, muss zusätzlich begründet werden, dass das Tie-Breaking mit einem gegebenen Ranking korrekt ist. Dafür wird zuerst das theoretische Konstrukt der idealen Grundverteilung nach Quelle [22] eingeführt.

Definition 6.1.1 (Ideale Grundverteilung)
Intuitiv ist die ideale Grundverteilung eine Sitzverteilung, nach der jede Partei genau so viele Sitze erhält, wie ihr Anteil an Stimmen ist. Dafür wird in Kauf genommen, dass im Allgemeinen auch Bruchteile von Sitzen vergeben werden müssen [22]. Die Sitze nach der Grundverteilung g_i einer Partei i werden also folgendermaßen berechnet [22]:

$$g_i = \frac{v_i}{v_{total}} \cdot n_{seats} = m_i + a_i \tag{6.3}$$

Dabei bezeichnen v_i die Stimmen von Partei i, v_{total} die Summe aller v_i und n_{seats} die Anzahl der zu verteilenden Sitze des Parlaments. g_i lässt sich in eine natürliche Zahl m_i und einen Rest a_i aufteilen, wobei $0 \leq a_i < 1$.

Da $\sum_{j=1}^{n_{parties}} g_j = n_{seats}$ gilt für jedes Sitzzuteilungsverfahren, das nur ganze Sitze verteilt:

$$\sum_{j=1}^{n_{parties}} m_j \leq n_{seats} \leq \left(\sum_{j=1}^{n_{parties}} m_j \right) + n_{parties} - 1 \tag{6.4}$$

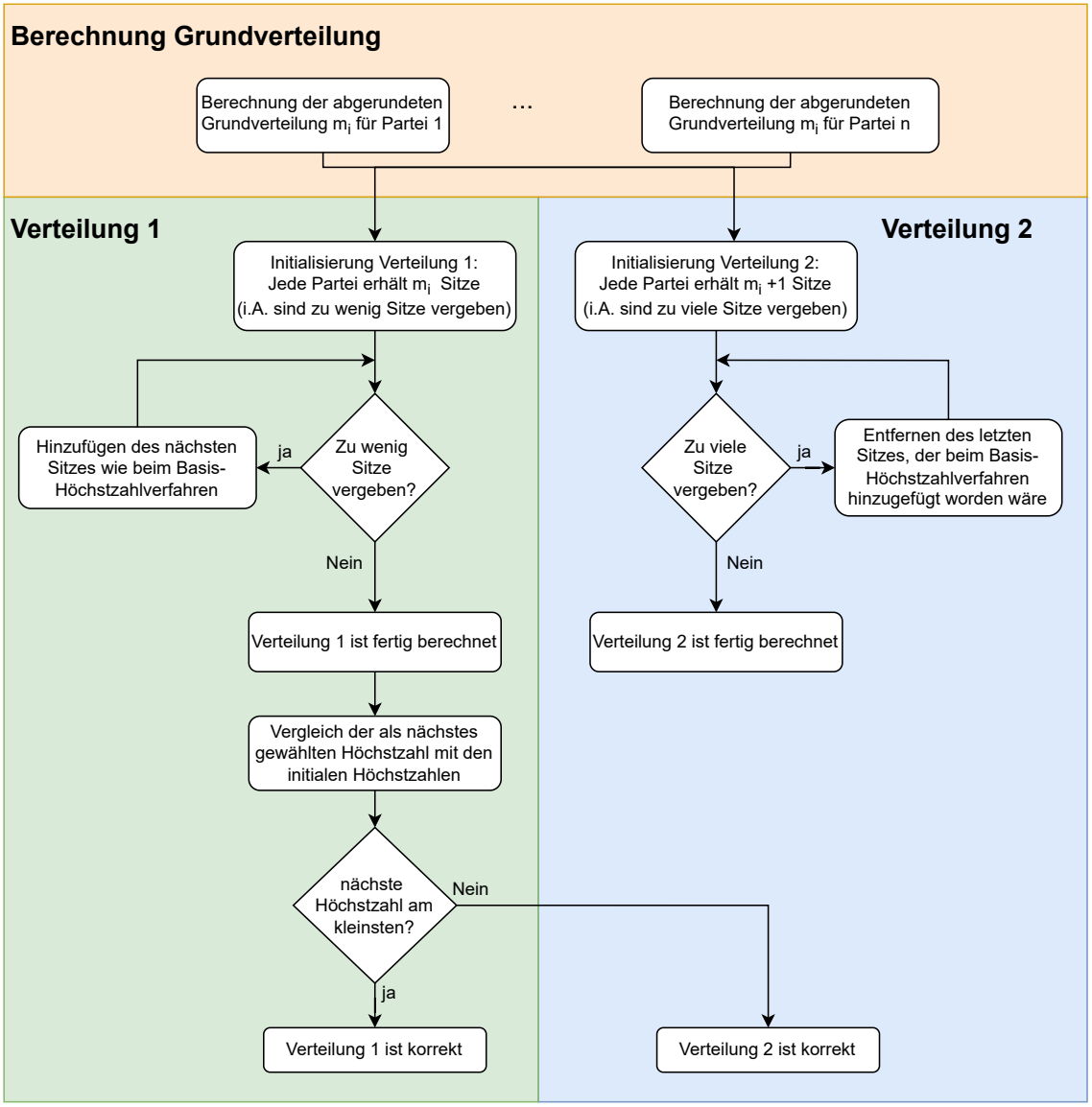


Abbildung 6.4: Ablauf des Höchstzahlverfahrens basierend auf der Grundverteilung: Zuerst wird die Grundverteilung für alle Parteien berechnet und davon ausgehend zwei mögliche Verteilungen ermittelt. Die erste Verteilung geht initial von zu wenig Sitzen aus und verteilt dann schrittweise Sitze nach dem Basis-Höchstzahlverfahren, während die zweite Verteilung von zu vielen Sitzen ausgeht und nach und nach Sitze entfernt. Am Ende wird durch Vergleich einiger bestimmter Höchstzahlen ermittelt, welche der Verteilungen korrekt ist.

Nun wird die ideale Grundverteilung in Zusammenhang mit der Sainte-Laguë-Sitzverteilung gebracht. Im Spezialfall, dass die ideale Grundverteilung eine gültige Sitzverteilung ist, das heißt dass alle g_i natürliche Zahlen sind, entspricht die Sitzverteilung nach Sainte-Laguë genau der idealen Grundverteilung [22].

Im allgemeinen Fall liegt die Sitzzahl der Parteien nur mit einer Wahrscheinlichkeit von unter $0,2\%$ außerhalb des Intervalls $[\lfloor g_i \rfloor, \lceil g_i \rceil]$. Dies wurde von Balinski et al. durch Simulation mit der Monte-Carlo-Methode gezeigt [6]. Allerdings soll das umgesetzte Verfahren immer korrekte Ergebnisse liefern und nicht nur mit großer Wahrscheinlichkeit. Es kann ab vier Sitzen vorkommen, dass die Sitzzahl einzelner Parteien von dem Intervall abweichen [6] und hierfür gibt es keine scharfe Obergrenze. Daher muss ein anderer Ansatz gefunden verwendet werden, als die Sitzverteilung in einem kleinen Intervall um die Grundverteilung festzulegen.

Theorem 8 (Relation der Sitze um die Grundverteilung)
Das Sainte-Laguë-Verfahren garantiert in der iterativen Formulierung als Höchstzahlverfahren folgendes: Bevor einer Partei i ihr $(m_i + 2)$-ter Sitz zugeteilt wird, müssen alle Parteien j bereits den m_j-ten Sitz erhalten haben [22]. Dabei bezeichnet m_i den abgerundeten Wert der idealen Grundverteilung.

Beweis Der Beweis folgt dem Vorgehen von Quelle [22]. Die Aussage gilt genau dann, wenn für alle Parteien i, j gilt: Die m_j-te Höchstzahl von jeder Partei j ist größter als die $(m_i + 2)$-te Höchstzahl von Partei i. Diese Eigenschaft wird im Folgenden gezeigt: Die erste Zeile des Beweises gilt trivialerweise, da m_i und m_j natürliche Zahlen sind.

$$m_j + m_i + 1 > 0 \tag{6.5}$$

$$\Leftrightarrow m_j(2 \cdot m_i + 3) > (m_i + 1)(2 \cdot m_j - 1) \tag{6.6}$$

$$\Leftrightarrow \frac{m_j}{2 \cdot m_j - 1} > \frac{m_i + 1}{2 \cdot m_i + 3} \tag{6.7}$$

$$\Rightarrow \frac{m_j + a_j}{2 \cdot m_j - 1} > \frac{m_i + a_i}{2 \cdot m_i + 3} \tag{6.8}$$

$$\Leftrightarrow \frac{g_j}{2 \cdot m_j - 1} > \frac{g_i}{2 \cdot m_i + 3} \tag{6.9}$$

$$\Leftrightarrow \frac{\frac{g_j \cdot v_{total}}{n_{seats}}}{2 \cdot m_j - 1} > \frac{\frac{g_i \cdot v_{total}}{n_{seats}}}{2 \cdot m_i + 3} \tag{6.10}$$

$$\Leftrightarrow \frac{v_j}{2 \cdot m_j - 1} > \frac{v_i}{2 \cdot m_i + 3} \tag{6.11}$$

Die letzte Zeile beschreibt die gewünschte größer-Beziehung der Höchstzahlen. ∎

Mit diesen Vorüberlegungen kann nun gezeigt werden, dass das beschriebene Verfahren korrekt ist. Dabei sind die Werte m_i des Höchstzahlverfahrens, das auf der Grundverteilung basiert, die abgerundeten Werte $\lfloor g_i \rfloor$ der idealen Grundverteilung.

Theorem 9 (Korrektheit des auf der Grundverteilung basierenden Höchstzahlverfahrens)
Das Evaluierungsprotokoll, das das Höchstzahlverfahren basierend auf der idealen Grundverteilung umsetzt und in Abschnitt 6.1.3 sowie in Abbildung 6.4 beschrieben wird, liefert immer eine korrekte Sitzverteilung nach Sainte-Laguë.

Beweis Aus Theorem 8 folgt, dass zwischen zwei Fällen unterschieden werden kann: Im ersten Fall erhält mindestens eine Partei i mindestens $(m_i + 2)$ Sitze. Dann folgt, dass alle Parteien j mindestens m_j Sitze zugewiesen bekommen. Außerdem folgt aus Gleichung 6.4, dass eine initiale Sitzverteilung, bei der jeder Partei m_j Sitze zugewiesen werden, nicht mehr Sitze als die Gesamtsitzzahl des Parlaments hat, und dass zum Erreichen dieser Gesamtzahl maximal $n_{parties}$ Sitze benötigt werden. Aus diesen Überlegungen ergibt sich die erste Verteilung des beschriebenen Verfahrens.

Im zweiten Fall erhält keine der Parteien $(m_i + 2)$ Sitze, das heißt jede Partei erhält maximal $(m_i + 1)$ Sitze. Gleichung 6.4 zeigt, dass wenn jeder Partei initial $(m_i + 1)$ Sitze zugeordnet werden, die Gesamtsitzzahl überschritten wird, aber dass maximal $n_{parties}$ Sitze abgezogen werden müssen, um die korrekte Sitzzahl zu erreichen. Daraus ergibt sich die zweite Verteilung des Verfahrens.

Das Hinzufügen und Abziehen der einzelnen Sitze erfolgt nach dem Höchstzahlverfahren, für das die Korrektheit bereits in Theorem 6 gezeigt wurde. Die einzige Möglichkeit, dass eine der Verteilungen falsch ist, besteht darin, dass die initiale Verteilung Sitze vergibt, die so nicht vergeben werden dürfen. Da die beiden Fälle komplementär sind, muss mindestens eine der Verteilungen korrekt sein. Daher wird nur überprüft, ob die erste Verteilung korrekt ist.

Die erste Verteilung ist nicht korrekt, wenn es mindestens eine Partei i gibt, die nach dem Sainte-Laguë-Verfahren keine m_j Sitze erhält. Auf das Höchstzahlverfahren übertragen bedeutet dies, dass es mindestens eine Höchstzahl h_i gibt, die Partei i den m_i-ten Sitz zuordnet, aber nicht zu den n_{seats} größten Höchstzahlen gehört. Das bedeutet insbesondere, dass h_i kleiner als die Höchstzahl ist, die als Nächstes zugeteilt werden würde. Daher werden im beschriebenen Verfahren alle Höchstzahlen, die den Parteien den m_i-ten Sitz zuweisen mit der Höchstzahl verglichen, die als Nächstes verteilt werden würde. Nur wenn es kein h_i gibt, das die Bedingung nicht erfüllt, ist die nächst-zuzuweisende Höchstzahl am kleinsten und die erste Verteilung ist korrekt. Dies wird so im Verfahren umgesetzt.

Da das Tie-Breaking ebenfalls beachtet wird, kann das Höchstzahlverfahren basierend auf der idealen Grundverteilung auf Wahlen in der Praxis angewandt werden. Das Tie-Breaking wird wie beim Basis-Höchstzahlverfahren durchgeführt. Es ist noch zu beachten, dass in Gleichung 6.5 immer ein <-Zeichen verwendet wird, sodass hier kein Gleichstand auftreten kann und in diesem Schritt kein Tie-Breaking nötig ist. ■

Außerdem ist das beschriebene Höchstzahlverfahren basierend auf der Grundverteilung sicher. Dies wird im folgenden Theorem gezeigt.

Theorem 10 (Sicherheit des auf der Grundverteilung basierenden Höchstzahlverfahrens)
Das Evaluierungsprotokoll, das das Höchstzahlverfahren basierend auf der idealen Grundverteilung umsetzt und in Abschnitt 6.1.3 sowie in Abbildung 6.4 beschrieben wird, erfüllt die Eigenschaften Tally Hiding, Accountability und Vote Privacy. Zusätzlich kann es als Teilprotokoll in weiteren Evaluierungsprotokollen verwendet werden.

Beweis Damit die drei Sicherheitseigenschaften erfüllt sind, müssen alle Voraussetzungen der Theoreme 1 und 2 gelten. Im Beweis von Theorem 7 wird gezeigt, dass die Voraussetzungen (a)-(d) von Theorem 1 und Voraussetzung (a) von Theorem 2 unabhängig vom konkreten Auswertungsprotokoll gelten. Daher muss nur gezeigt werden, dass Bedingung (e) aus Theorem 1 und Bedingung (b) aus Theorem 2 erfüllt sind.

Zunächst werden das Tally Hiding und die Vote Privacy gezeigt. Wie in Theorem 7 wird die Voraussetzung für das Tally Hiding so verschärft, dass auch das Endergebnis nicht entschlüsselt werden darf, damit das Protokoll als Teilprotokolle in andere Evaluierungsfunktionen eingebaut werden kann. Diese Eigenschaft ist erfüllt, da keine Zwischenergebnisse oder Ergebnisse entschlüsselt werden, sodass keine Informationen preisgegeben werden.

Auch Voraussetzung (b) aus Theorem 2 für die Accountability ist erfüllt, da über die Dokumentation auf dem Bulletin Board überprüft werden kann, dass die Teilschritte korrekt berechnet werden und dass der Output eines Teilprotokolls immer korrekt als Input des nächsten Teilprotokolls verwendet wird. ■

Benchmarking des Verfahrens

Das Verfahren wird sowohl sequentiell als auch parallelisiert gebenchmarkt. Beide Ergebnisse sind in Abbildung 6.5 dargestellt. Bei der sequentiellen Variante werden die Berechnung der Grundverteilung, der ersten Verteilung und der zweiten Verteilung nach Abbildung 6.4 nacheinander ausgeführt. Die sequentielle Variante berechnet die Grundverteilung parallel für jede Partei und auch die erste und zweite Verteilung werden parallel berechnet. Beide Varianten sind für große Sitzzahlen und verhältnismäßig wenige Parteien schneller als das Basis-Höchstzahlverfahren und eignen sich daher gut für die Anwendung in den Evaluierungsfunktionen, die in den nächsten Kapiteln entwickelt werden.

Für die Verwendung in weiteren Evaluierungsprotokollen wie der Umsetzung der Wahl des norwegischen Parlaments und der Bundestagswahl wird der sequentielle Ansatz verwendet. Bei entsprechender Hardware, die genügend parallele Prozesse ausführen kann, kann in diesen Verfahren auch der parallele Ansatz verwendet werden.

6.1.4 Weitere Implementierungen des Höchstzahlverfahrens

Zusätzlich zu den vorgestellten Verfahren wurden noch weitere Varianten umgesetzt. Einerseits wurde eine unverschlüsselte Form das Höchstzahlverfahrens implementiert. Diese kann verwendet werden, wenn alle Parameter bekannt sind, da in diesem Fall auch die Sitzverteilung bekannt ist. Da die Berechnung unverschlüsselt stattfindet, muss weniger auf Effizienz geachtet werden, denn die unverschlüsselten Operationen können sehr schnell ausgeführt werden.

Zum anderen werden weitere spezifische Varianten des Sainte-Laguë-Verfahrens benötigt. Dies sind vor allem Anpassungen an der Art des Tie-Breakings, wenn die Wahlverfahren dies verlangen. Diese Änderungen betreffen die Wahl des norwegischen Parlaments und die Bundestagswahl und werden bei der Umsetzung der Wahlverfahren in den nächsten Kapiteln beschrieben.

6.2 Umsetzung der Wahl des House of Commons

Die Wahl des britischen Parlaments folgt einem eher einfachen Verfahren und sie wird als Beispiel verwendet, um zu zeigen, dass mit Ordinos auch einfache politische Wahlen umgesetzt werden können, die die Anforderungen der Praxis erfüllen. Daher wird die Wahl des House of Commons in

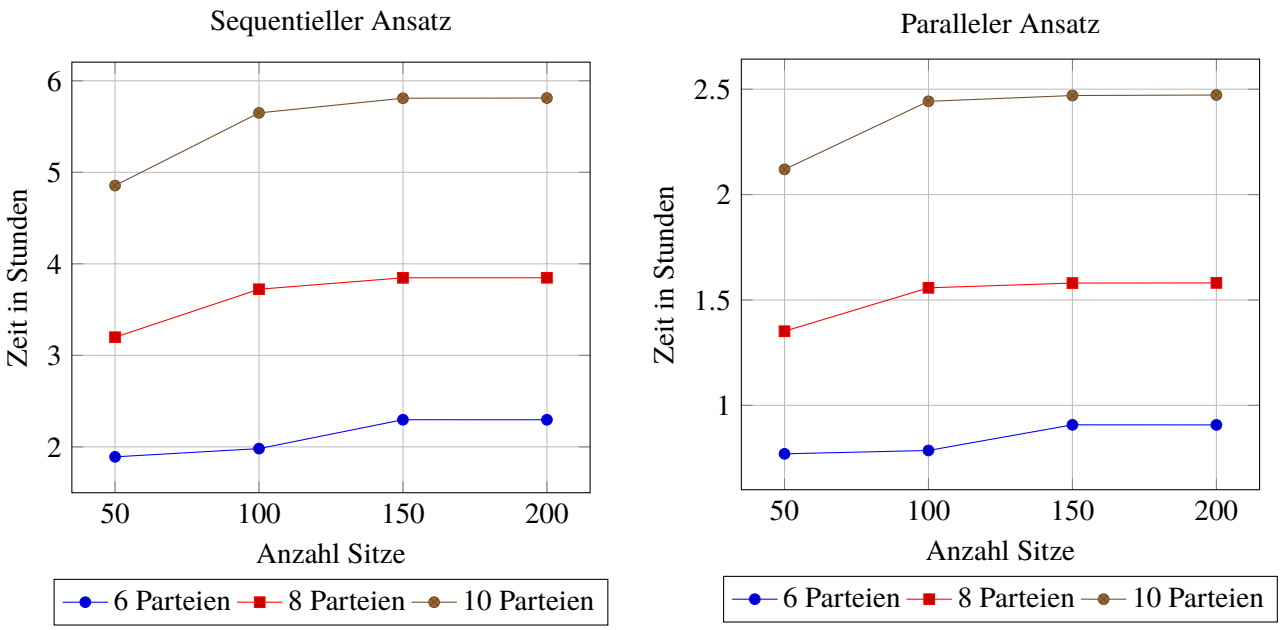


Abbildung 6.5: Benchmark für das auf der Grundverteilung basierende Höchstzahlverfahren: Links sind die Schritte sequentiell ausgeführt und rechts wurde so parallelisiert, wie es in Abbildung 6.4 dargestellt ist. Für die Verwendung des Verfahrens in weiteren Evaluierungsfunktionen wird der sequentielle Ansatz verwendet. Für große Sitzzahlen und wenig Parteien ist diese Umsetzung des Höchstzahlverfahrens deutlich schneller als das Basis-Höchstzahlverfahren. Die Laufzeit ist im Allgemeinen linear und die Knicke in den Graphen kommen durch die Vergleiche mit unterschiedlicher Bitlänge zustande.

Ordinos im Wesentlichen so umgesetzt wie es durch die allgemeinen Änderungen im vorherigen Kapitel angedeutet wurde. In Abbildung 6.6 ist ein Überblick über die Auswertung der Wahl des britischen Unterhauses sowie alle dafür benötigten Ergänzungen zu sehen.

In der Setup Phase kann entschieden werden, wie bei Bedarf ein Gleichstand aufgelöst wird. Wenn Zufall gewählt wird, kann die Generierung der zufälligen Ranks von Ordinos übernommen werden. Daraufhin wird pro Wahlkreis gewählt und die Stimmen werden auf ein gemeinsames Bulletin Board gelegt und anschließend nach Wahlkreis sortiert und aggregiert.

Nun folgt das eigentliche Evaluierungsprotokoll. Bei den weiteren Wahlverfahren wird nur noch dieses beschrieben, da der erste Teil identisch ist. Die gesamte Auswertungsfunktion sowie das Ergebnis sind nach Wahlkreisen aufgeteilt. Bei der Wahl des britischen Parlaments muss pro Wahlkreis der Gewinner bestimmt werden. In Ordinos existiert bereits eine Evaluierungsfunktion, um die k Personen mit den meisten Stimmen zurückzugeben [31]. Allerdings können hier bei Gleichstand zwischen Kandidaten unter Umständen mehr als k Kandidaten zurückgegeben werden.

Wenn man einen Sieger mit einfacher Mehrheit finden möchte und Tie-Breaking beachtet, gibt es zwei Möglichkeiten: Man kann entweder aus den Stimmen pro Kandidat und den Ranks pro Kandidat Punkte zu berechnen und der Kandidat mit den meisten Punkten ist Sieger oder man kann zuerst alle möglichen Sieger finden und zwischen diesen den Gleichstand auflösen. Durch Benchmarks wurde gezeigt, dass die erste Variante schneller ist. Daher wurde die in Ordinos implementierte

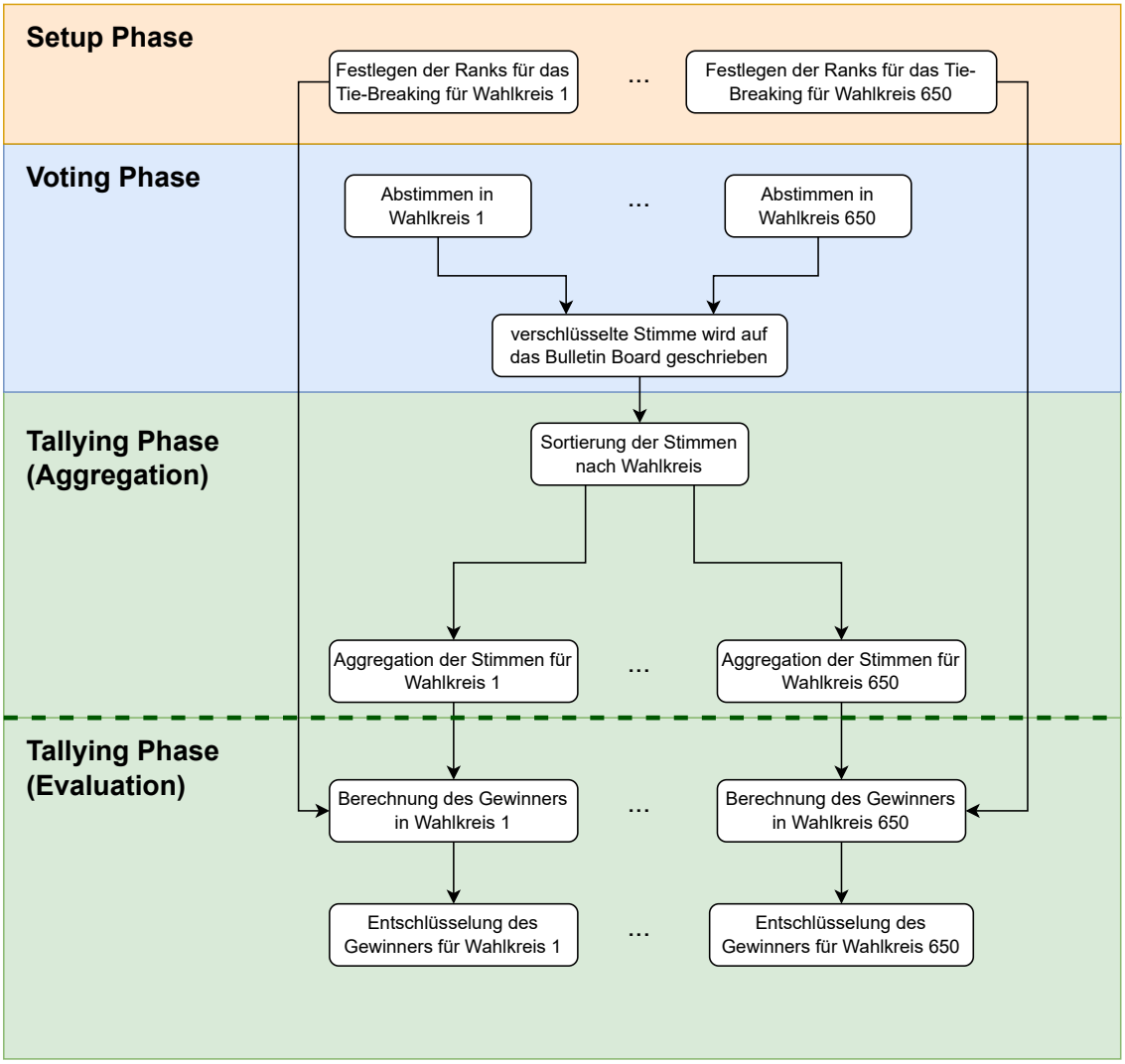


Abbildung 6.6: Überblick über die Umsetzung der Wahl des House of Commons: Hier sind alle wesentlichen Ergänzungen in den einzelnen Phasen für die Wahl des House of Commons. (Abgesehen von der Evaluation sind die Phasen nicht vollständig dargestellt, sondern enthalten nur die Änderungen.) Die Auswertung ist nach Wahlkreisen getrennt und die Berechnungen für die Wahlkreise können parallel stattfinden. Nur das Bulletin Board wird zentral verwendet und enthält die gesamte Dokumentation aller Wahlkreise.

Funktion nicht verwendet und stattdessen der Wahlkreissieger folgendermaßen aus den Punkten berechnet: Es wird linear durch die Liste an Punkten durchgegangen und der Index des aktuellen Maximums gespeichert. Bei jedem Element der Liste wird das neue Element mit dem aktuellen Maximum verglichen und das aktuelle Maximum wird bei Bedarf angepasst. Da das Aktualisieren des Maximums nur mit Additionen und Multiplikationen möglich ist, wird so pro Element nur ein größer-gleich-Vergleich benötigt [34].

Nachdem der Wahlkreissieger berechnet wurde, kann der Index sofort entschlüsselt werden und auf dem Bulletin Board nachgeschlagen werden, zu welchem Kandidaten dieser Index gehört. Um das Ergebnis geordnet darzustellen, werden die Sieger der einzelnen Wahlkreise zusammengefasst und diese Menge entspricht den neuen Abgeordneten des Parlaments.

6.2.1 Korrektheit und Sicherheit

Die beschriebene Umsetzung mit Ordinos liefert trivialerweise immer das korrekte Ergebnis, da sie sich sehr stark an der tatsächlichen Umsetzung der Wahl orientiert. Außerdem ist das beschriebene Protokoll sicher.

Theorem 11 (Sicherheit der Umsetzung der Wahl des House of Commons)
Die in Abbildung 6.6 und Abschnitt 6.2 beschriebene Umsetzung der Wahl des House of Commons in Ordinos erfüllt die Eigenschaften Tally Hiding, Vote Privacy und Accountability.

Beweis Der Beweis hierfür ist analog zum Beweis von Theorem 7. Nach der gleichen Begründung muss für die Sicherheit nur gezeigt werden, dass Voraussetzung (e) aus Theorem 1 und Voraussetzung (b) aus Theorem 2 erfüllt sind.

In dem beschriebenen Protokoll werden keine Zwischenergebnisse entschlüsselt. Die einzigen Informationen, die entschlüsselt werden, sind die Sieger jedes Wahlkreises. Dies stellt kein Problem dar, da jeder Wahlkreissieger ein Teil des Ergebnisses, das heißt der Parlamentsverteilung, ist. Jeder Kandidat darf nur in einem Wahlkreis antreten. Wenn also ein Kandidat als Abgeordneter ins Parlament einzieht, ist öffentlich bekannt, dass dieser in seinem Wahlkreis gewonnen hat. Damit sind alle Voraussetzungen für Theorem 1 erfüllt und somit die Eigenschaften Tally Hiding und Vote Privacy bewiesen.

Accountability ist ebenfalls gewährleistet, da durch die Zero-Knowledge-Beweise und die Dokumentation auf dem Bulletin Board überprüft werden kann, ob jeder Trustee die korrekten Daten als Input verwendet und die Berechnungen korrekt durchführt. Da genügend ehrliche Trustees für das Entschlüsseln des Ergebnisses zur Verfügung stehen, kann auch hier keine Manipulation unbemerkt durchgeführt werden. ■

Benchmarks

Da die Auswertungen pro Wahlkreis getrennt stattfinden, wird zuerst betrachtet, wie lange die Berechnung des Gewinners eines Wahlkreises benötigt. Dafür wird eine realistische maximale Wählergröße von 111.000 Wähler angenommen [18]. Da die Laufzeit nur von der Anzahl an zur

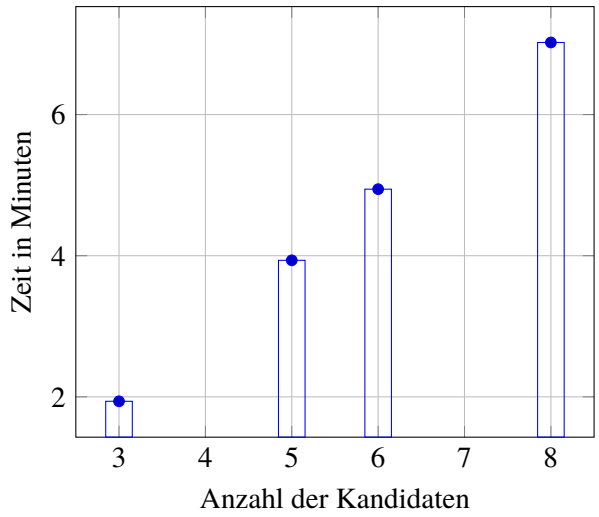


Abbildung 6.7: Benchmark für das Evaluierungsprotokoll der Wahl des House of Commons. Die Zeiten beziehen sich je auf einen Wahlkreis und im Diagramm sind die Werte für die wichtigsten statistischen Größen der Verteilung der Kandidaten pro Wahlkreis dargestellt: Die meisten Wahlkreise haben zwischen drei und acht Kandidaten, der Modus liegt bei fünf Kandidaten und im Durchschnitt traten 5, 1 Kandidaten an.

Wahl stehenden Kandidaten abhängig ist, werden in Abbildung 6.7 Benchmarks mit den wichtigsten statistischen Größen durchgeführt: Über 95% aller Wahlkreise haben zwischen drei und acht Kandidaten, der Modus liegt bei 5 Kandidaten und das arithmetische Mittel bei 5, 1 Kandidaten [21].

Für den Durchschnitt von 5, 1 Kandidaten wurde mit linearer Interpolation ein Wert von vier Minuten und zwei Sekunden errechnet. Da in 650 Wahlkreisen gewählt wird und die Zeit pro Wahlkreis linear in Abhängigkeit von der Anzahl der Kandidaten ist, wurde mit diesem Wert die Gesamtlaufzeit errechnet. Außerdem wird die Berechnung parallelisiert. Da der Vergleichsrechner vier Kerne hat, kann die Auswertung von vier Wahlkreisen gleichzeitig berechnet werden. Daher beträgt die Laufzeit für die Wahl des House of Commons knapp elf Stunden.

Obwohl für diese Berechnungen die Werte der letzten Wahl verwendet wurden, ist nicht zu erwarten, dass die Laufzeit über unterschiedliche Wahlen merklich variiert. Zwar können bei zukünftigen Wahlen in einigen Wahlkreisen mehr oder weniger Kandidaten antreten, doch vermutlich ändert sich der Mittelwert über alle 650 Wahlkreise nur gering. Da die Auswertung eines Wahlkreises unter zehn Minuten beträgt, ist die Gesamtlaufzeit hauptsächlich deshalb so lange, da bei der Wahl viele Wahlkreise verwendet werden. Mit entsprechender Hardware, die mehr parallele Berechnungen ausführen kann, könnte die Laufzeit daher deutlich verringert werden.

6.3 Umsetzung der Wahl des Storting

In Abschnitt 4.9 wird beschrieben, wie die Auswertung einer Wahl des norwegischen Parlaments abläuft. Dieses Kapitel zeigt nun, wie dieses Evaluierungsverfahren in Ordinos umgesetzt werden kann. Bei dem vorliegenden Protokoll wird das Verfahren wieder in die drei Schritte ”Berechnung

der Bezirkssitze", "Verteilung der Ausgleichssitze" und "Zuordnung der Ausgleichssitze auf die Bezirke" aufgeteilt. Dennoch gibt es innerhalb der einzelnen Schritte teils erhebliche Abweichungen von der offiziellen Auswertung, wie sie in Abschnitt 4.9 beschrieben ist. Diese Abweichungen sind nötig, um einerseits die Umsetzung mit Tally Hiding zu ermöglichen und andererseits ein effizientes Verfahren aufzustellen. Da diese Veränderungen nichts am Ergebnis einer Wahl ändern dürfen, wird bei nichttrivialen Änderungen gezeigt, dass die Korrektheit weiterhin gewährleistet ist.

In den folgenden Abschnitten wird die Umsetzung der einzelnen Schritte mit Ordinos detailliert beschrieben. Anschließend wird gezeigt, dass alle Sicherheitseigenschaften, die Ordinos verlangt, erfüllt sind und es wird durch Benchmarks nachgewiesen, dass das Verfahren effizient genug ist, um in der Realität angewandt werden zu können. Der gesamte Ablauf des Evaluierungsprotokolls ist in Abbildung 6.8 schematisch dargestellt.

6.3.1 Berechnung der Bezirkssitze

Um die Bezirkssitze zu berechnen, wird das modifizierte Basis-Höchstzahlverfahren verwendet, wie es in Abschnitt 6.1.2 beschrieben wurde. Um es hier anzuwenden, muss noch das Tie-Breaking hinzugefügt werden. Hierbei wird bei Gleichstand die Partei bevorzugt, die mehr Stimmen erhält und wenn die Parteien gleich viele Stimmen haben, wird durch Zufall entschieden.

In diesem Schritt ist besonders, dass es im Allgemeinen mehr Parteien als Sitze gibt: Die Anzahl der antretenden Parteien liegt zwischen 17 und 23 und ist im Durchschnitt 19, während die Anzahl an zu verteilenden Sitzen zwischen drei und 19 liegt und im Durchschnitt acht ist [27]. Dies kann für das Tie-Breaking genutzt werden. Damit Gleichstand zwischen Parteien vorliegt, müssen deren Höchstzahlen gleich sein. Dafür gibt es zwei Fälle: Im ersten Fall sind mehrere gleiche Höchstzahlen in der gleichen Zeile der Tabelle. Da hier die Nenner der Höchstzahlen gleich sind, folgt, dass sowohl die Anzahl an Stimmen als auch die Anzahl an bereits zugeordneten Sitzen der Parteien gleich sind. In diesem Fall muss durch Zufall entschieden werden. Im zweiten Fall sind mehrere gleiche Höchstzahlen in unterschiedlichen Zeilen der Tabelle. Da dann die Nenner unterschiedlich sind, müssen die Anzahl der Stimmen und die Anzahl der bereits zugeordneten Sitze pro Partei unterschiedlich sein. In diesem Fall hat die Partei mit mehr Stimmen auch mehr Sitze. Daher ist es äquivalent, ob im Zweifel nach der Anzahl an Stimmen oder der Anzahl an Sitzen entschieden wird.

Da die Anzahl an Sitzen relativ gering ist und es somit oft nur wenige Iterationen gibt, wird die Anzahl an Sitzen als Grundlage für das Tie-Breaking verwendet. Diese werden mit den zufälligen Ranks kombiniert, um beide Fälle abzudecken.

Die Berechnung der einzelnen Bezirke wird parallel ausgeführt, aber alle Operationen innerhalb einer Sainte-Laguë-Auswertung müssen sequentiell durchgeführt werden. Dadurch ist dieser Schritt zeitlich beschränkt durch den Bezirk, dessen Auswertung am längsten dauert. Hierfür kommen am Beispiel der letzten Wahl [27] zwei Bezirke in Frage: Der Bezirk Oslo hat mit 19 Sitzen die meisten Bezirkssitze zu verteilen, während der Bezirk Hordaland die Sitze unter den meisten Parteien (22) verteilt. Benchmarks mit den korrekten Sitz- und Parteienanzahlen sowie realen Größenordnungen von Stimmen haben gezeigt, dass die Auswertung in Oslo am längsten dauert, weshalb dies als Obergrenze für den aktuellen Schritt verwendet wird.

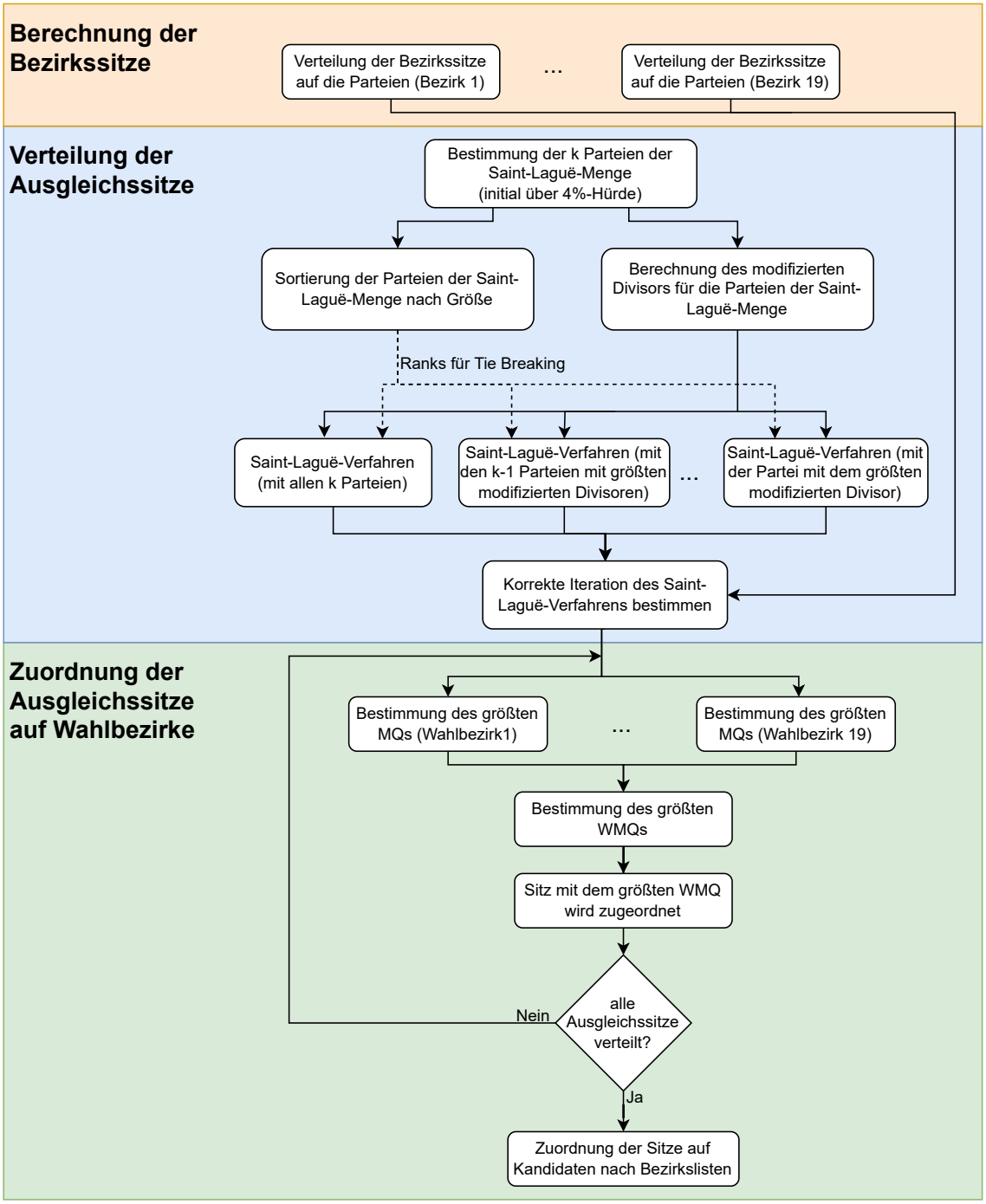


Abbildung 6.8: Schematische Darstellung der Umsetzung der norwegischen Wahl: Im Gegensatz zur theoretischen Beschreibung wird vieles so umgewandelt, dass es parallelisierbar ist. Um dies zu realisieren, mussten vor allem beim mittleren Schritt einige Berechnungen hinzugefügt werden.

6.3.2 Verteilung der Ausgleichssitze

Ausgleichssitze werden nur an Parteien verteilt, die landesweit über 4% der Stimmen erhalten [26]. Daher kann in diesem Schritt eine Vereinfachung durchgeführt werden und es wird entschlüsselt, welche Parteien im Parlament Sitze gewinnen. Dies sind die Parteien, die im vorherigen Schritt mindestens einen Bezirkssitz erhalten haben oder mindestens vier Prozent der Stimmen erhalten haben, um in diesem Schritt Ausgleichssitze bekommen zu können. Im weiteren Verlauf werden nur diese Parteien berücksichtigt. Da Parteien, die Bezirkssitze erhalten, aber unter der 4%-Hürde sind, keine weiteren Sitze durch die folgenden Schritte erhalten dürfen, werden ihre Stimmen vorläufig auf null gesetzt.

Die ursprüngliche Verteilung der Ausgleichssitze hat das Problem, dass die Berechnung sequentiell stattfindet: Zuerst wird mit den Parteien der Sainte-Laguë-Menge eine Verteilung bestimmt und überprüft, ob diese valide ist, das heißt, ob keine Partei weniger Sitze erhält als ihr nach den Bezirkssitzen zustehen. Wenn die Verteilung für eine Partei invalide ist, wird diese aus der Sainte-Laguë-Menge entnommen, ihr werden alle Bezirkssitze zugeteilt und der Vorgang wird wiederholt. Bei der letzten Wahl des norwegischen Parlaments im Jahr 2021 haben zehn Parteien Sitze gewonnen, weshalb dieser Vorgang bis zu neun mal wiederholt werden müsste. Früheres Abbrechen der Iterationen wäre wegen des Tally Hidings nicht möglich, denn es soll pro Partei geheim bleiben, ob sie ihre Sitze nur durch Bezirkssitze oder auch durch Ausgleichssitze erhalten hat.

Da eine einzelne Iteration fast sechs Stunden benötigt, ist es nicht möglich, neun sequentielle Iterationen zu nutzen, sondern es muss optimiert werden, sodass die einzelnen Iterationen parallel durchgeführt werden. Die Herausforderung dabei ist, dass nach dem ursprünglichen Algorithmus erst am Ende einer Iteration die Parameter für die nächste Iteration feststehen, das heißt die Sainte-Laguë-Menge sowie daraus folgend die Anzahl an zu verteilenden Sitzen. Also wird eine Möglichkeit benötigt, zu Beginn festzulegen, welche Parteien in jeder Iteration zur Sainte-Laguë-Menge gehören. Daraus kann direkt berechnet werden, wie viele Sitze in der jeweiligen Iteration zu verteilen sind.

Ein naiver Ansatz wäre, alle möglichen Kombinationen, welche Parteien in der Sainte-Laguë-Menge enthalten sein könnten, durchzugehen und die Verteilung jeweils parallel ausrechnen zu lassen. Anschließend könnte bestimmt werden, welche der Kombinationen für die Wahl die Korrekte ist. Dieser Ansatz hat allerdings den Nachteil, dass die Anzahl an parallel berechneten Verteilungen nach dem modifizierten Höchstzahlverfahren in etwa der Mächtigkeit der Potenzmenge der Sainte-Laguë-Menge entspricht. Für die letzte Wahl des norwegischen Parlaments mit zehn Parteien, die Sitze erhalten haben [27], wären dafür etwa $10! = 3.628.800$ parallele Prozesse nötig. Doch dies ist mit der vorgegebenen Hardware keinesfalls machbar.

Der final gewählte Ansatz hingegen kommt mit nur zehn parallelen Prozessen aus und läuft folgendermaßen ab: Zuerst wird für jede Partei i der folgende Quotient $\tilde{d}_i$ ausgerechnet, wobei v_i die Stimmen und $\tilde{s}_i$ die Summe der Bezirkssitze bezeichnen:

$$\tilde{d}_i = \frac{v_i}{\tilde{s}_i} \tag{6.12}$$

Anschließend werden die Parteien nach den Quotienten $\tilde{d}_i$ aufsteigend sortiert und die Sainte-Laguë-Mengen für die einzelnen Iterationen bestimmt. Für die erste Iteration enthält die Sainte-Laguë-Menge alle Parteien, die Sitze im Parlament erhalten; in der nächsten Iteration wird die Partei mit dem kleinsten $\tilde{d}_i$ herausgenommen usw. bis für die letzte Iteration nur noch die Partei mit dem

größten $\tilde{d}_i$ in der Sainte-Laguë-Menge übrig bleibt. Somit können die Elemente der Sainte-Laguë-Menge vor Beginn der Verteilung bestimmt werden, weshalb die Iterationen parallel ausgeführt werden können.

Als Vorbereitung für die einzelnen Berechnungen der Sainte-Laguë-Verteilung müssen noch Tie-Breaking-Ranks berechnet werden. In der norwegischen Wahl wird bei Gleichstand ein Sitz der Partei mit den meisten Stimmen zugeordnet, danach wird durch Zufall entschieden. Also müssen alle Parteien der Sainte-Laguë-Menge nach ihren Stimmanzahlen sortiert werden. Diese Sortierung wird mit den Zufallspermutationen für das Tie-Breaking verrechnet, um alle Fälle abzudecken. Da nur eine der Iterationen in die finale Sitzverteilung einfließt und es somit durch den Zufall keine systematische Bevorzugung oder Benachteiligung einzelner Parteien geben kann, können die berechneten Tie-Breaking-Ranks für alle Iterationen verwendet werden. Sie werden zu Beginn parallel zu der Berechnung und Sortierung der $\tilde{d}_i$s aufgestellt.

Da die Umwandlung des sequentiellen Ansatzes in den parallelen Ansatz nicht intuitiv ist, wird im Folgenden gezeigt, dass beide Ansätze gleichwertig sind, sodass man mit dem entwickelten parallelen Ansatz immer zu korrekten Ergebnissen kommt. Für diese Begründung wird ein Hilfssatz benötigt, der eine Mindestanzahl an Sitzen der Parteien über der 4%-Hürde definiert.

Theorem 12 (Mindestsitzzahl nach der Sainte-Laguë-Verteilung)
Im zweiten Schritt bei der Auswertung der Wahl des norwegischen Parlaments erhält jede Partei, die mindestens vier Prozent der Stimmen erhalten hat, mindestens zwei Sitze nach der Sainte-Laguë-Verteilung.

Beweis In Quelle [30] gibt Kopfermann eine untere Schranke dafür an, welchen Stimmanteil eine Partei mindestens haben muss, um mindestens eine gegebene Anzahl an Sitzen nach dem klassischen Sainte-Laguë-Verfahren zu erhalten. Aus dieser Formel folgt, dass eine Partei mindestens zwei Sitze erhält, wenn gilt [30]:

$$\frac{v_i}{v_{total}} \geq \frac{2,5}{n_{seats} - 0,5 \cdot n_{parties} + 1} \tag{6.13}$$

Da der Stimmanteil für jede der betrachteten Parteien bei mindestens vier Prozent liegt, folgt, dass maximal 25 Parteien über dieser Hürde liegen können und die Formel lässt sich in folgende Aussage umformen:

$$n_{seats} \geq 74 \tag{6.14}$$

Da bei der Wahl des norwegischen Parlaments 169 Sitze verteilt werden, ist diese Aussage erfüllt. Somit wurde gezeigt, dass jede der Parteien mit dem klassischen Sainte-Laguë-Verfahren mindestens zwei Sitze erhält. Daher ist das klassische Sainte-Laguë-Verfahren äquivalent zum modifizierten Sainte-Laguë-Verfahren und lässt sich auf die Auswertung der Wahl des norwegischen Parlaments übertragen.

Außerdem ist die Schranke in Gleichung 6.14 umso kleiner, je weniger Parteien an der Sainte-Laguë-Sitzverteilung teilnehmen. ■

Die Schranke von 74 Sitzen ist außerdem so klein, dass sie auch dann erfüllt ist, wenn einige Parteien unter der 4%-Hürde Sitze erhalten und daher nicht mit einer initialen Verteilung von 169 Sitzen begonnen wird.

Mit dieser Vorarbeit lässt sich nun die Korrektheit des Verfahrens zeigen.

Theorem 13
Die Verteilung der Ausgleichssitze, wie sie in Abschnitt 6.3.2 und Abbildung 6.8 beschrieben wird, ist korrekt, da das Verfahren äquivalente Ergebnisse zur ursprünglichen Definition in Abschnitt 4.9.2 und Abbildung 4.3 liefert.

Beweis Bei beiden Verfahren werden (in Abhängigkeit von der Anzahl der Parteien im Parlament) gleich viele Iterationen des Höchstzahlverfahrens ausgeführt. Der parallele Ansatz ist richtig, wenn die Sainte-Laguë-Mengen in den jeweiligen Iterationen gleich sind und die korrekte Iteration für die endgültige Verteilung ausgewählt wird.

Zuerst wird gezeigt, dass die Sainte-Laguë-Mengen bei den beiden Ansätzen gleich sind. Nach dem Divisorverfahren, das äquivalent zum Höchstzahlverfahren ist, existiert ein Divisor d, sodass für alle Parteien i gilt:

$$\left\lceil \frac{v_i}{d} \right\rceil = s_i \tag{6.15}$$

Da in diesem Schritt alle Parteien über der 4%-Hürde liegen, erhält jede Partei mindestens zwei Sitze. Dies wird in Abschnitt 12 begründet. Daher sind das klassische Höchstzahlverfahren und das modifizierte Höchstzahlverfahren äquivalent. Da das Divisorverfahren mit dem klassischen Höchstzahlverfahren äquivalent ist, gilt die Gleichung 6.15 auch für das modifizierte Höchstzahlverfahren. Allerdings sind die finalen Sitze pro Partei s_i noch nicht bekannt, dafür aber die Summe der Bezirkssitze $\tilde{s}_i$. Daraus lässt sich für die Parteien, die letztendlich durch die Sainte-Laguë-Verteilung zusätzliche Sitze erhalten, folgende Gleichung umformen:

$$d \approx \frac{v_i}{s_i} \leq \frac{v_i}{\tilde{s}_i} = \tilde{d}_i \tag{6.16}$$

Für die Parteien hingegen, denen mehr Bezirkssitze zustehen, als sie nach der Sainte-Laguë-Verteilung erhalten würden, gilt:

$$d \approx \frac{v_i}{s_i} \geq \frac{v_i}{\tilde{s}_i} = \tilde{d}_i \tag{6.17}$$

Daraus folgt, dass für den Quotienten d des Divisorverfahrens gilt:

$$\max_{i \notin \text{Sainte-Laguë-Menge}} \tilde{d}_i \leq d \leq \min_{i \in \text{Sainte-Laguë-Menge}} \tilde{d}_i \tag{6.18}$$

Wenn also nun für jede Iteration das kleinste $\tilde{d}_i$ aus der Sainte-Laguë-Menge entfernt wird, tritt in einer der Iterationen der Fall ein, dass die Parteien so in der Sainte-Laguë-Menge enthalten sind, dass die letzte Gleichung gilt. In diesem Fall würde man mit dem Divisorverfahren zum richtigen Ergebnis kommen und den Parteien der Sainte-Laguë-Menge würden so viele Sitze zugeteilt werden, dass das $\tilde{d}_i$ etwa dem d entspricht. (Eine Gleichheit ist wegen der Rundung der Sitze nicht notwendig.) Der exakte Divisor d bleibt aber unbekannt und wird nicht benötigt, da statt dem Divisorverfahren für die Implementierung das äquivalente Höchstzahlverfahren verwendet wird. Daraus folgt, dass die endgültige Verteilung in einer der Iterationen enthalten ist.

Nun muss gezeigt werden, dass die korrekte Verteilung ausgewählt wird. Für jede der berechneten Sainte-Laguë-Verteilungen kann überprüft werden, ob sie valide ist, das heißt, ob jede Partei der Sainte-Laguë-Menge mindestens ihre Bezirkssitze erhält. Im Allgemeinen gilt das für die Verteilungen mehrerer Iterationen. Nach der ursprünglichen Konstruktion, in der iterativ für jede

Runde Parteien entfernt werden, ist die Verteilung die Finale, die unter allen validen Sitzverteilungen am meisten Parteien in der Sainte-Laguë-Menge enthält. Da in jeder Runde eine unterschiedliche Anzahl an Parteien in der Sainte-Laguë-Menge sind, ist die finale Verteilung eindeutig.

In der Implementierung wird dafür pro Runde ein Index auf ”eins” oder ”null” gesetzt, je nachdem ob die Iteration valide ist oder nicht. Indem von dem aktuellen Index der Wert aus der vorherigen Runde abgezogen wird, kann festgestellt werden, bei welcher Runde der Wechsel von invalide zu valide stattfindet. Diese Runde ist die korrekte und wird als Ergebnis zurückgegeben. ■

6.3.3 Zuordnung der Ausgleichssitze auf die Bezirke

In diesem Schritt wird zuerst berechnet, welche Parteien wie viele Ausgleichssitze haben. Dass jedem Wahlbezirk ein Ausgleichssitz zur Verfügung steht, ist bekannt.

Ziel ist es nun in einer Schleife nach und nach die 19 Ausgleichssitze der Parteien auf die Wahlbezirke zu verteilen. Dabei muss in jedem Schleifendurchlauf der größte gewichtete Mandatsquotient nach Formel 4.8 berechnet werden. Dies geschieht in zwei Schritten. Zuerst wird in jedem Wahlbezirk parallel der größte ungewichtete Mandatsquotient nach Formel 4.7 berechnet, da die Gewichtung nur abhängig von dem Bezirk, aber nicht von der Partei ist. Allerdings wird der Mandatsquotient schon hier auf ”null” gesetzt, wenn die Partei keinen Überhang hat oder der Bezirk keinen freien Ausgleichssitz mehr hat. Falls ein Gleichstand herrscht, wird das Tie-Breaking durch Zufall gelöst, und das größte Element wird wieder berechnet, indem einmal durch alle Parteien linear iteriert wird.

Im zweiten Schritt werden die Mandatsquotienten zwischen den einzelnen Bezirken verglichen. Dabei wird jeder Mandatsquotient gewichtet und der größte Quotient ermittelt. Das Tie-Breaking ist für diesen Schritt zweistufig definiert: Zuerst wird der Bezirk mit mehr Wählern bevorzugt und wenn noch immer Gleichstand herrscht, durch Zufall entschieden. Für dieses System können die Tie-Breaking-Ranks relativ schnell ausgerechnet werden, da die Anzahl an Wählern pro Bezirk unverschlüsselt auf dem Bulletin Board steht. Also können die Wählerzahlen unverschlüsselt verglichen werden und die Ranks können den Parteien direkt zugeordnet werden und nur bei einem Gleichstand müssen für wenige Bezirke die Ranks durch verschlüsselte Zufallspermutationen erweitert werden.

Nachdem der größte gewichtete Mandatsquotient gefunden wurde, wird der entsprechenden Partei in dem entsprechendem Bezirk ein weiterer Sitz zugeordnet, der Überhang der Partei wird dekrementiert und es wird gespeichert, dass dieser Bezirk keinen Ausgleichssitz mehr benötigt. Anschließend werden die Mandatsquotienten erneut berechnet und der Vorgang wird wiederholt.

Bestimmung des Wahlergebnisses

Nun steht fest, wie viele Sitze jede Partei in jedem Wahlbezirk erhält. Diese Information ist das Ergebnis und kann entschlüsselt werden. Zur besseren Darstellung des Ergebnisses müssen die Sitze auf Kandidaten verteilt werden. Wenn eine Partei in einem Wahlkreis k Sitze erhält, ziehen die ersten k Kandidaten der Landesliste dieser Partei als Abgeordnete ins Parlament ein.

Korrektheit und Sicherheit

Das in den letzten Abschnitten und in Abbildung 6.8 beschriebene Protokoll zur Umsetzung der Wahl des norwegischen Parlaments liefert immer die richtigen Ergebnisse. Für den Schritt "Verteilung der Ausgleichssitze" wurde dies explizit gezeigt. Für die anderen beiden Schritte ist die Korrektheit trivialerweise erfüllt, da die Berechnungen sehr nahe an der ursprünglichen Formulierung des Verfahrens in Kapitel 4.9 angelehnt sind. Außerdem ist das entwickelte Protokoll sicher, wie im nächsten Abschnitt gezeigt wird.

Theorem 14
Das beschriebene Protokoll für die Umsetzung der Wahl des norwegischen Parlaments mit Ordinos erfüllt die Eigenschaften Tally Hiding, Vote Privacy und Accountability.

Beweis Die Begründung der Sicherheitseigenschaften folgt genau dem Schema der vorherigen Sicherheitsbegründungen. Daher muss nur noch Voraussetzung (e) aus Theorem 1 gezeigt werden, die besagt, dass neben dem Ergebnis keine weiteren Informationen veröffentlicht werden. Das Ergebnis ist für dieses Protokoll die finale Parlamentsverteilung, das heißt die Anzahl an Sitzen, die jede Partei in jedem Bezirk erhält.

In diesem Protokoll findet aber eine weitere Entschlüsselung statt, da die Parteien entschlüsselt werden, die über der 4%-Hürde sind oder mindestens einen Bezirkssitz erhalten. Damit das Tally Hiding gewährleistet bleibt, muss die entschlüsselte Information Teil des Ergebnisses sein. Dies ist erfüllt, denn jede Partei, die mindestens vier Prozent der Stimmen erhält, bekommt garantiert mindestens einen Sitz im Parlament. Dies folgt direkt aus Theorem 12. Gleiches gilt für Parteien, die bereits einen Bezirkssitz erhalten haben. Außerdem ist es für Parteien, die weder vier Prozent der Stimmen noch einen Bezirkssitz erhalten haben, unmöglich, in der weiteren Auswertung Sitze zu erhalten. Daher entspricht die Menge der entschlüsselten Parteien genau den Parteien, die in der endgültigen Sitzverteilung mindestens einen Sitz bekommen. Da die Sitzverteilung als Ergebnis öffentlich ist, sind auch die Parteien, die Sitze erhalten, bekannt. ■

Benchmarks

In Abbildung 6.9 sind die Benchmarks der Wahl des norwegischen Parlaments dargestellt. Dabei sind die Zeiten pro Auswertung in die Verteilung der Bezirkssitze, die Verteilung der Ausgleichssitze und die Zuordnung der Ausgleichssitze auf die Bezirkssitze aufgeteilt. In der Mitte sind dabei die Zeiten für die Berechnung des Wahlergebnisses der letzten Wahl zu sehen. Im Folgenden wird beschrieben, wie die Berechnungszeiten zukünftiger Wahlen von diesem abweichen können.

Im ersten Schritt werden parallel pro Wahlkreis die Bezirkssitze verteilt. Daher ist die Laufzeit durch den Wahlkreis beschränkt, der am meisten Sitze zu verteilen hat. Die Sitze, die jedem Wahlkreis zugeordnet sind, können für die Betrachtung dieser Arbeit nicht verändert werden. (In der Realität werden die Sitze alle acht Jahre neu vergeben, doch hier sind keine großen Änderungen zu erwarten, da die Vergabe abhängig von relativ konstanten Größen wie der Fläche und der Bevölkerung ist [26].) Daher kann die Zeit für den ersten Schritt nur variieren, wenn mehr oder weniger Parteien antreten. Doch auch hier wurde durch Benchmarks gezeigt, dass die Abweichung durch zwei antretende Parteien mehr oder weniger nur minimal ist. Daher kann die Zeit für den ersten Schritt unabhängig von der Anzahl der Parteien als konstant betrachtet werden. Dass die Zeit für die Berechnung nur von

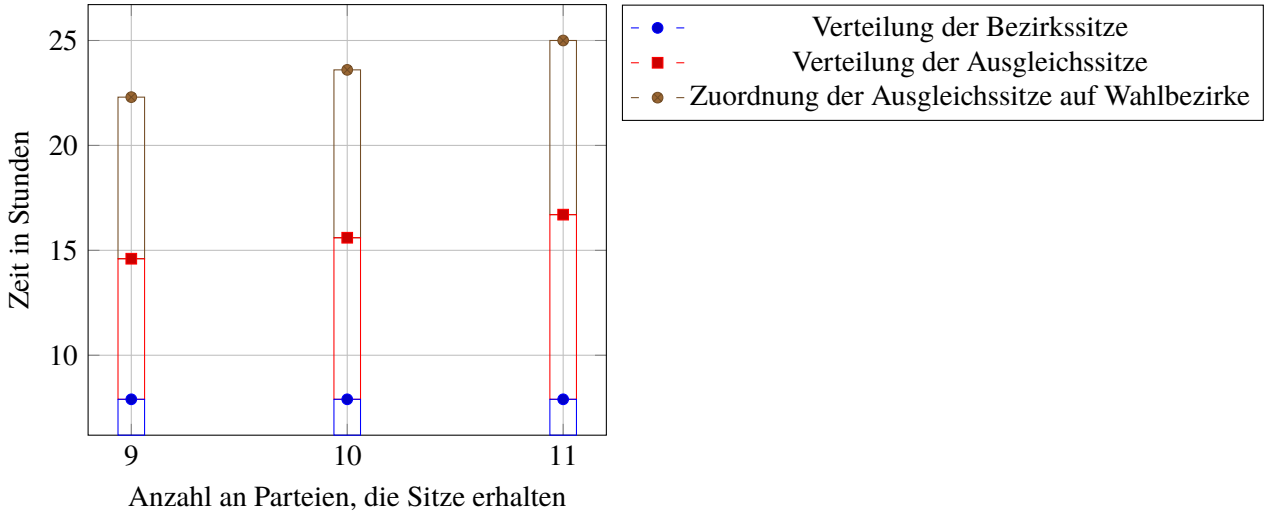


Abbildung 6.9: Benchmarks für die Auswertung der Wahl des norwegischen Parlaments. Bei der Verteilung der Bezirkssitze sind bei unterschiedlichen Wahlen sehr ähnliche Zeiten zu erwarten, während die Auswertungszeiten der anderen beiden Schritte davon abhängen, wie viele Parteien ins Parlament einziehen.

bekannten Größen, aber nicht von den Stimmen der Wähler abhängig ist, ist bei einer Auswertung mit Tally Hiding zu erwarten, da keine aus den Wählerstimmen abhängige Information entschlüsselt wird.

Im zweiten Schritt werden die Ausgleichssitze verteilt und im Gegensatz zum ersten Schritt kann es hier abhängig von der Verteilung der Wählerstimmen zu unterschiedlichen Auswertungszeiten kommen. Der Grund dafür ist, dass zu Beginn dieses Schrittes entschlüsselt wird, welche Parteien Sitze im Parlament gewinnen. Diese Entschlüsselung reduziert die Laufzeit deutlich, da beispielsweise mit den Daten der diesjährigen Wahl für die weiteren Schritte statt mit über 20 Parteien nur noch mit zehn Parteien gerechnet werden muss. Wenn weniger Parteien ins Parlament einziehen, müssen die Sitze unter weniger Parteien verteilt werden, wodurch Zeit gespart wird. Da eine Partei ins Parlament kommt, wenn sie mindestens einen Bezirkssitz oder mindestens vier Prozent der Stimmen erhalten hat, ist die Berechnungszeit abhängig von der Verteilung der Wählerstimmen. Wenn die Wähler nur wenige (Volks-) Parteien wählen, ziehen weniger Parteien ins Parlament ein und die Auswertung benötigt kürzere Zeit. Sind die Wählerstimmen jedoch so gleichmäßig unter den Parteien verteilt, dass möglichst viele Parteien Sitze erhalten, wird mehr Zeit für die Verteilung der Sitze unter den Parteien benötigt. In Extremfällen zieht nur eine Partei ins Parlament ein oder jeder Bezirkssitz geht an eine andere Partei, sodass das Parlament 150 verschiedene Parteien enthält. Doch diese Randfälle sind in der Praxis unwahrscheinlich, daher wurde neben den zehn Parteien, die in der letzten Wahl Sitze gewonnen haben, das Verfahren zusätzlich noch für neun und elf Parteien mit Sitzen gebenchmarkt.

Auch der letzte Schritt ist abhängig von der Anzahl der Parteien, die ins Parlament einziehen. Dies folgt aus der Begründung für den vorherigen Schritt. In der Abbildung ist zu sehen, dass die Zuordnung der Ausgleichssitze auf die Bezirkssitze einen großen Teil der Auswertungszeit einnimmt. Da bereits am Ende des zweiten Schrittes die Sitze pro Partei feststehen und nun nur noch die Ausgleichssitze auf die Wahlbezirke verteilt werden müssen, könnte man sich überlegen,

diesen Schritt wegzulassen und entweder die Ausgleichssitze oder alle Sitze durch nationalweite Listen anstatt durch auf Bezirke aufgeteilte Listen vergeben. Ein ähnlicher Vorschlag wurde bereits in Quelle [26] gemacht, da die Verteilung auf die Wahlbezirke relativ kompliziert ist und im Allgemeinen nicht exakt zum Wahlergebnis in diesem Bezirk passt.

Mit entsprechender Hardware wäre es auch möglich, die Auswertungszeit weiter zu verkürzen. Vor allem bei der Verteilung der Ausgleichssitze durch das Höchstzahlverfahren basierend auf der Grundverteilung kann noch deutlich parallelisiert werden. Die Einzelheiten dazu wurden bereits im entsprechenden Kapitel beschrieben.

6.4 Umsetzung der Wahl des Deutschen Bundestags

Die einzelnen Komponenten der Bundestagswahl werden wie vom Bundeswahlleiter beschrieben [46] und in Abbildung 4.4 gezeigt umgesetzt. Doch innerhalb einer Komponente werden die Verfahren oft angepasst, um die Anforderungen der Sicherheitseigenschaften, vor allem des Tally Hidings, und der Effizienz umzusetzen. Die Auswertung der Bundestagswahl ist das komplexeste Verfahren dieser Arbeit und beinhaltet die meisten Teilschritte. Doch es konnten einige Algorithmen und Ideen aus vorherigen Wahlen wiederverwendet werden. Dies betrifft vor allem das Sainte-Laguë-Verfahren, das Rechnen mit Brüchen und die Bestimmung des Maximums aus einer Menge. Diese Algorithmen werden daher hier nur kurz beschrieben. Im Folgenden wird für jede Komponente erklärt, wie sie in Ordinos umgesetzt wird. In Abbildung 4.4 ist ebenfalls erkennbar, ob die Schritte parallel oder sequentiell ausgeführt werden. Im Allgemeinen werden die Schritte parallel ausgeführt, in denen die gleiche Berechnung unabhängig pro Partei oder pro Bundesland ausgeführt wird. Teils wäre noch mehr Parallelisierung innerhalb einzelner Komponenten möglich, doch diese wird für die Benchmarks nicht beachtet, da sonst zu viele parallele Prozesse vorliegen würden.

6.4.1 Berechnung der Direktmandate

Die Direktmandate pro Wahlkreis werden analog zu der Auswertung der Wahl des House of Commons berechnet. Zwar wird hier bei Gleichstand zwischen mehreren Kandidaten durch Zufall entschieden, doch die Berechnung der Tie-Breaking-Ranks findet bereits in der Setup-Phase statt und ist daher für diesen Schritt nicht relevant. Da die Direktmandate aber nur ein Zwischenergebnis der Auswertung sind, dürfen sie nicht entschlüsselt werden.

In der letzten Bundestagswahl standen in den Wahlkreisen zwischen fünf und 18 Kandidaten für das Direktmandat zur Wahl [8]. Für das Benchmarking wird die durchschnittliche Anzahl von $11,2$ Kandidaten pro Wahlkreis [8] verwendet. Die Berechnung des Kandidaten mit den meisten Stimmen ist linear in der Anzahl der Kandidaten. Daher wird davon ausgegangen, dass die Annäherung durch den Mittelwert der Kandidaten über die 299 Wahlkreise etwa zu den gleichen Laufzeiten führt, wie wenn für jeden Wahlkreis mit der tatsächlichen Kandidatenanzahl gebenchmarkt werden würde. Für die Parallelisierung wird davon ausgegangen, dass vier Prozesse parallel ausgeführt werden können.

6.4.2 Bestimmung der großen Parteien

Als große Parteien sind die Parteien definiert, die im weiteren Verlauf der Berechnung beachtetet werden müssen. Sie werden folgendermaßen berechnet: Für jede Nicht-Minderheitspartei wird getestet, ob sie mindestens fünf Prozent der Zweitstimmen erhalten hat und ob ihr mindestens drei Direktmandate zustehen. Diese beiden Indikatoren werden Oder-verknüpft, indem sie addiert werden und anschließend getestet wird, ob die Summe größer als "null" ist. Der daraus entstehende Indikator ist genau dann "eins", wenn die Partei mindestens fünf Prozent der Stimmen oder drei Direktmandate erhalten hat. Anschließend wird der Indikator entschlüsselt und die Partei wird bei Bedarf den großen Parteien hinzugefügt. Minderheitsparteien werden immer den großen Parteien hinzugefügt und es wird nicht getestet, ob sie die anderen beiden Bedingungen erfüllen. Nun wird für jede Partei entschlüsselt, ob sie eine große Partei ist, oder nicht.

6.4.3 Erste Oberverteilung

Bei der ersten Oberverteilung ist die Besonderheit, dass alle Eingabe-Parameter für das Sainte-Laguë-Verfahren, also die Bevölkerungsverteilung und die Gesamtsitzzahl, bekannt sind. Aus diesem Grund kann jeder das Ergebnis, das heißt die Sitze pro Bundesland, im Klartext berechnen. Daher ist hier eine Verschlüsselung nicht notwendig und die Sitze pro Bundesland werden durch das unverschlüsselte Sainte-Laguë-Verfahren berechnet.

6.4.4 Erste Unterverteilung

In diesem Schritt werden die Sitze pro Bundesland unter den Parteien verteilt. Im Allgemeinen ist die Anzahl der zu verteilenden Sitze hier deutlich höher als die Anzahl der Parteien. Beispielsweise wurden in der letzten Bundestagswahl zwischen fünf und 127 Sitze auf durchschnittlich sechs Parteien verteilt. Daher wird für jedes Bundesland das Höchstzahlverfahren, das auf der idealen Grundverteilung basiert, aus Abschnitt 6.1.3 verwendet. Dies beinhaltet bereits das Tie-Breaking und gibt das Ergebnis verschlüsselt zurück, sodass es ohne weitere Anpassungen verwendet werden kann.

Für manche Bundesländer mit wenig Sitzen könnte auch das Basis-Höchstzahlverfahren aus Abschnitt 6.1.2 verwendet werden. Doch für die Gesamtlaufzeit ist dies irrelevant, da die Laufzeit durch das Bundesland mit den meisten Sitzen beschränkt ist.

6.4.5 Berechnung des Mindestsitzanspruchs

Die Berechnung des Mindestsitzanspruchs erfolgt parallel für jede Partei.

Dabei wird zuerst für jedes Bundesland die Mindestsitzzahl der Partei berechnet. Nach Formel 4.9 gibt es für die Mindestsitzzahl zwei Fälle:

$$s_i^{min} = \begin{cases} s_i^{dir}, \text{ wenn } s_i^{kon} \leq s_i^{dir} \\ \left\lceil \frac{s_i^{dir}+s_i^{kon}}{2} \right\rceil, \text{ wenn } s_i^{kon} > s_i^{dir} \end{cases} \tag{6.19}$$

Algorithmus 6.5 Berechnung der Mindestsitzzahl für eine Partei und ein Bundesland

procedure CalculateMinSeats($E_{pk}(seatsContingent)$, $E_{pk}(seatsDirect)$)
  $E_{pk}(diff) = E_{pk}(seatsContingent) -_e E_{pk}(seatsDirect)$
  $E_{pk}(roundedDiff) =$ FloorDivision($E_{pk}(diff) +_e 1, E_{pk}(2), \lceil \frac{a_{max}}{2} \rceil$)
  $E_{pk}(roundedMean) = E_{pk}(seatsDirect) +_e E_{pk}(roundedDiff)$
  $E_{pk}(conGt) =$ gt($E_{pk}(seatsContingent), E_{pk}(seatsDirect)$)
  $E_{pk}(minSeats) =$ ifThenElse($E_{pk}(conGt), E_{pk}(roundedMean), E_{pk}(seatsDirect)$)
  return $E_{pk}(minSeats)$
end procedure

Nun wird eine Funktion für den gerundeten Mittelwert im zweiten Fall definiert. Anstatt die Summe aus Sitzkontingent und Anzahl der Direktmandate des Bundeslandes durch zwei zu dividieren, wird nur die Differenz durch zwei geteilt. Dies hat den Vorteil, dass für die Division weniger Iterationen benötigt werden. Da in Algorithmus 6.4 bereits ein Protokoll für die abgerundete Division entwickelt wurde, wird dieses Protokoll für die hier benötigte aufgerundete Division wiederverwendet. Da durch zwei geteilt wird, kann zwischen folgenden beiden Fällen entschieden werden, wobei d die Differenz zwischen dem Sitzkontingent und der Anzahl der Direktmandate ist:

$$\left\lceil \frac{d}{2} \right\rceil = \begin{cases} \frac{d+1}{2}, \text{ d ist ungerade} \\ \frac{d}{2}, \text{ d ist gerade} \end{cases} = \left\lfloor \frac{d+1}{2} \right\rfloor \tag{6.20}$$

Nun kann die aufgerundete Differenz wie in Formel 6.20 erklärt berechnet werden. Anschließend wird die Anzahl der Direktmandate, die die Partei in dem Bundesland erhalten hat, auf das Ergebnis summiert.

Bei der beschriebenen Berechnung von $\left\lceil \frac{s_i^{dir}+s_i^{kon}}{2} \right\rceil$ ist das Ergebnis nur sinnvoll, wenn $s_i^{kon} \geq s_i^{dir}$ gilt. Ansonsten ist die Differenz negativ und daher ist das Ergebnis der aufgerundeten Division falsch. Für das Verfahren ist dies aber kein Problem, da im Fall, dass $s_i^{kon} \leq s_i^{dir}$ das Ergebnis der Division nicht benötigt wird. Der gesamte Algorithmus für die Berechnung der Mindestsitzzahl ist in Algorithmus 6.5 beschrieben.

Anschließend wird diese Mindestsitzzahl pro Bundesland aufsummiert und mit der Summe der Sitzkontingente verglichen. Das Maximum dieser beiden Werte ist der Mindestsitzanspruch für diese Partei.

6.4.6 Zweite Oberverteilung

In der zweiten Oberverteilung werden die Gesamtsitze erhöht. Würde man diese Gesamtsitzzahl iterativ erhöhen, müsste man die vorherigen Schritte bei jeder Iteration durchführen und zusätzlich das Tie-Breaking beachten. Daher wird stattdessen der Algorithmus verwendet, der in Abschnitt 4.10.4 beschrieben ist. Zuerst wird wie in Abschnitt 4.10.4 der Divisor bestimmt.

Um das Minimum d_{ohne} der Menge aus Formel 4.10 zu bestimmen, werden die Elemente der Menge als Brüche dargestellt und das Minimum wird wie bei der Iteration des Basis-Höchstzahlverfahrens linear bestimmt. Hier ist kein Tie-Breaking nötig, da nur der Wert des Minimums, aber nicht die zugehörige Partei relevant ist. Damit Zähler und Nenner der Brüche ganze Zahlen sind, wird jeder Bruch aus Formel 4.10 mit zwei erweitert.

Das viertkleinste Element der Menge D aus Formel 4.11 wird ähnlich bestimmt. Hierbei wird vier mal über die Elemente von D iteriert, wobei nach den ersten drei Durchläufen das gefundene Minimum aus der Menge entfernt wird, indem ein Index auf "null" gesetzt wird. Daher muss bei allen Iterationen außer der Ersten durch einen Gleichheitstest geprüft werden, dass das neue Minimum keines der Werte ist, die auf "null" gesetzt werden. Außerdem muss der Fall beachtet werden, dass das erste Element der Menge in vorherigen Runden auf "null" gesetzt wurde. Dafür wird ein Boolean verwendet, sodass durch Multiplikation und Addition das erste Element, das nicht "null" ist, als initiales Minimum definiert ist.

Nun wird mit dem bekannten Divisor d, der das Minimum der beiden berechneten Divisoren ist, die Sainte-Laguë-Verteilung berechnet. Um die Division zu umgehen, wird statt dem Divisorverfahren das Höchstzahlverfahren verwendet, für das im Folgenden eine weitere Variante eingeführt wird.

Höchstzahlverfahren mit gegebenem Divisor

Diese Variante ist für den Fall optimiert, dass der Divisor und die Anzahl an Stimmen pro Partei bekannt sind, aber sowohl die Anzahl an Sitzen pro Partei als auch die Gesamtzahl an Sitzen unbekannt sind. Nach Theorem 3 ist bei der Umformung zwischen Höchstzahlverfahren und Divisorverfahren die Eigenschaft erfüllt, dass der Divisor im Intervall $(2 \cdot h(n_{seats} + 1), 2 \cdot h(n_{seats})]$ liegt. Hierbei bezeichnet $h(n_{seats})$ die n_{seats}-größte Höchstzahl, wenn n_{seats} Sitze zu verteilen sind. Daraus folgt für jede Höchstzahl, der ein Sitz zugeordnet wird, dass der Divisor kleiner oder gleich das Doppelte dieser Höchstzahl ist. Hingegen ist bei Höchstzahlen, denen kein Sitz zugeordnet wird, der Divisor größer als das Doppelte dieser Höchstzahlen. Wenn eine Partei in der finalen Sitzverteilung k Sitze erhält, gilt für den Divisor:

$$\frac{2 \cdot v}{2 \cdot k + 1} < d \leq \frac{2 \cdot v}{2 \cdot k - 1} \tag{6.21}$$

Aus diesen Eigenschaften lässt die Methode `testSeats` generieren, um für eine gegebene Sitzzahl k für eine Partei zu testen, ob diese korrekt ist. Diese vergleicht den Divisor mit der Höchstzahl $\frac{2 \cdot v}{2 \cdot k - 1}$ und entschlüsselt das Ergebnis des Vergleichs. Ist der Divisor größer als die Höchstzahl, wird "False" zurückgegeben, im anderen Fall "True". Die beiden Rückgabewerte liegen dabei unverschlüsselt vor. Die endgültige Anzahl an Sitzen, die in diesem Schritt berechnet wird, ist die größte Zahl k, für die "True" zurückgegeben wird.

Daraus lässt sich ein Algorithmus für das Höchstzahlverfahren mit gegebenem Divisor entwickeln. Ein einfacher Ansatz, um zu bestimmen, an welcher Stelle der Rückgabewert der Methode `testSeats` von "True" auf "False" wechselt, ist aufsteigend alle möglichen Sitzzahlen durchzugehen. Allerdings kommen in der Praxis relativ hohe Sitzzahlen vor (z. B. 206 Sitze für die SPD) [46], sodass dieses Verfahren recht ineffizient ist. Daher wurde die deutlich schnellere Methode gewählt, zuerst in Hunderterschritten zu testen, in welchem Hundertstelbereich sich die gesuchte Sitzzahl befindet und anschließend in diesem Bereich eine binäre Suche durchzuführen. Dieser Algorithmus ist als Pseudocode in Abbildung 6.6 dargestellt. Tie-Breaking ist in diesem Schritt nicht notwendig, da im Falle eines Gleichstandes alle betroffenen Parteien den zusätzlichen Sitz erhalten [46]. Dies ist unproblematisch, da die Gesamtanzahl an Sitzen nicht festgelegt ist.

Algorithmus 6.6 Höchstzahlverfahren mit gegebenem Divisor

```
procedure HÖCHSTZAHLVERFAHREN(E_pk(votes), E_pk(divisor))
    for p ∈ parties do
        isLess = True
        hundreds = 0
        while isLess do                          // erste Approximation für Hunderter
            isLess =TESTSEATS(hundreds, E_pk(divisor), E_pk(votes))
            if isLess then
                hundreds+ = 100
            end if
        end while
        lowerBound = hundreds − 100
        upperBound = hundreds
        while lowerBound < upperBound do         // exakte Sitzzahl mit binärer Suche
            k = ⌊lowerBound + 1/2(upperBound − lowerBound)
            isLess = TESTSEATS(k, E_pk(divisor), E_pk(votes_party))
            if isLess then
                lowerBound = k
            else
                upperBound = k
            end if
            if lowerBound + 1 = upperBound then
                break
            end if
        end while
        seatsParty = lowerBound
    end for
end procedure
```

Als weitere Verbesserung könnte man auch das Vorwissen nutzen, das man über die einzelnen Parteien hat. Als Grundlage hierfür kann man frühere Wahlergebnisse oder Umfragen zur Wahl nutzen. Auch wenn die Sitzverteilung zwischen unterschiedlichen Legislaturen schwanken kann, bekommen Minderheitsparteien wie der Südschleswigscher Wählerverband (SSW) im Allgemeinen nur wenige Sitze und Volksparteien wie die CDU im Allgemeinen über hundert Sitze. Diese Verbesserung wurde allerdings nicht implementiert, da die verwendete binäre Suche recht effizient ist und es schwierig ist, eine solche Verbesserung ohne Bias zu testen, wenn die aktuellen Wahlergebnisse bekannt sind.

Zusätzlich müssen noch die bis zu drei unausgeglichenen Überhangmandate beachtet werden. In diesem Fall werden einer Partei mehr Sitze zugewiesen als ihr nach dem modifizierten Höchstzahlverfahren zustehen. Die unausgeglichenen Überhangmandate sind allerdings in dem Mindestsitzanspruch der Parteien enthalten. Daher wird die Methode `testSeats` so modifiziert, dass auch dann "True" zurückgegeben wird, wenn die zu testende Sitzzahl k kleiner oder gleich dem Mindestsitzanspruch der Partei ist.

Die Berechnung der Divisoren am Anfang erfolgt sequentiell, doch das Höchstzahlverfahren mit gegebenem Divisor wird für jede Partei parallel berechnet.

6.4.7 Zweite Unterverteilung

In diesem Schritt wird die Anzahl der Sitze pro Partei auf die einzelnen Bundesländer verteilt. Ausgenommen sind hiervon die unausgeglichenen Überhangmandate. Dieser Schritt verläuft analog zur ersten Unterverteilung, indem das Höchstzahlverfahren, das auf der Grundverteilung basiert, verwendet wird und die Berechnungen pro Partei parallelisiert werden. Der Unterschied zur ersten Unterverteilung ist, dass die Sitze unter Bundesländern anstatt unter Parteien verteilt werden. Da es mehr Bundesländer als Parteien mit Sitzen gibt und da mehr Sitze verteilt werden als in der ersten Unterverteilung, ist die Laufzeit größer.

6.4.8 Berechnung der verbleibenden Überhänge

Dieser Schritt ist ähnlich zu dem berechnen des Divisors der zweiten Oberverteilung. Der Hauptunterschied besteht darin, dass hier Tie-Breaking beachtet werden muss, da der Index des Maximums und nicht der Wert relevant ist. Zunächst wird für jede Partei die Menge L_i nach Formel 4.12 aufgestellt. Dabei werden die Brüche so erweitert, dass Zähler und Nenner je eine natürliche Zahl sind. Anschließend wird mit Hilfe der Vergleichsoperationen der Bruch-Klasse das Minimum bestimmt und dem zugehörigen Bundesland ein Sitz zugeteilt. Dann wird das gefundene Minimum auf Null gesetzt. Dieser Schritt wird für jede Partei dreimal ausgeführt. Bei den letzten beiden Durchgängen muss das zweitkleinste beziehungsweise drittkleinste Element der Menge gefunden werden. Dafür wird bei der Bestimmung des Minimums jeweils durch Gleichheits-Tests ausgeschlossen, dass das aktuelle Element ”null” ist.

Anschließend wird aus den letzten beiden Schritten errechnet, wie viele Sitze jede Partei in jedem Bundesland erhält und diese Information wird entschlüsselt.

6.4.9 Feststellung des Ergebnisses

Im letzten Schritt werden aus den Direktmandaten und der Verteilung nach den Zweitstimmen die Abgeordneten des Bundestages bestimmt, ohne dass bekannt ist, ob ein Kandidat seinen Sitz durch ein Direktmandat oder einen Listenplatz erhalten hat. Dafür wird für jede Partei für jedes Bundesland eine Liste aller Kandidaten angelegt. Diese Liste wird als Abgeordnetenliste bezeichnet. Dort wird jedem Kandidaten ein Wert zugewiesen, der ”eins” oder ”null” ist, je nachdem, ob der Kandidat in den Bundestag einzieht oder nicht.

Initial werden alle diese Werte auf ”null” gesetzt. Zuerst werden die Direktmandate verteilt. Die Direktmandate sind als ähnliche Liste mit Einsen und Nullen gespeichert, und können daher direkt in die Abgeordnetenliste übertragen werden. Zusätzlich wird ausgerechnet, wie viele Kandidaten durch die Direktmandate in diesem Bundesland für diese Partei ein Mandat erhalten.

Anschließend werden die Mandate nach Zweitstimmen verteilt. Dieser Teil wird nur ausgeführt, wenn es sich um eine große Partei handelt. Diese Anzahl ist bekannt, da sie im letzten Schritt entschlüsselt wurde. Dabei wird von oben nach unten durch die Abgeordnetenliste iteriert und getestet, ob die Summe der Abgeordneten der Anzahl der Listenplätze entspricht. Ist dies nicht erreicht, wird dem aktuellen Abgeordneten ein Mandat gegeben. Wenn dieser Abgeordnete bereits ein Direktmandat erhalten hat, wird nichts verändert. Als letztes wird die Abgeordnetenliste entschlüsselt, sodass bekannt ist, welche Abgeordnete ins Parlament einziehen.

Dieses Vorgehen wird für alle Bundesländer und für alle Parteien durchgeführt. Dabei kann zwischen den Bundesländern parallelisiert werden. Theoretisch wäre es auch möglich, innerhalb eines Bundeslandes zu parallelisieren, doch dafür stehen nicht genügend parallele Prozesse zur Verfügung.

Als Letztes werden die Direktmandate der kleinen Parteien entschlüsselt. Eine kleine Partei kann Sitze erhalten, wenn sie weniger als drei Direktmandate gewinnt. (Bei mehr Direktmandaten handelt es sich um eine große Partei.) In diesem Fall werden der Partei nur die Direktmandate und keine weiteren Sitze durch die Sainte-Laguë-Verteilung zugeteilt.

6.4.10 Korrektheit und Sicherheit

Die beschriebene Umsetzung der Bundestagswahl mit Ordinos ist korrekt. Grund dafür ist, dass in den meisten Fällen die Umsetzung relativ direkt dem Vorgehen der offiziellen Auswertung des Bundeswahlleiters folgt [46]. Für die Teilverfahren, bei denen dies nicht der Fall ist, wie bei der Sainte-Laguë-Auswertung, wurde die Korrektheit bereits im Laufe dieser Arbeit gezeigt. Außerdem ist auch die Auswertung der Bundestagswahl sicher.

Theorem 15
Das in Kapitel 6.4 beschriebene Protokoll für die Umsetzung der Wahl des Deutschen Bundestages mit Ordinos erfüllt die Eigenschaften Tally Hiding, Vote Privacy und Accountability.

Beweis Um die drei Sicherheitseigenschaften zu zeigen, müssen alle Voraussetzungen aus den Theoremen 1 und 2 erfüllt sein. Wie bei vorherigen Beweisen beschrieben sind die meisten dieser Voraussetzungen unabhängig vom konkreten Evaluierungsprotokoll bereits in Ordinos erfüllt und es muss nur Voraussetzung (e) aus Theorem 1 gezeigt werden. Diese besagt, dass keine Information außer das Ergebnis entschlüsselt werden darf. Im Protokoll für die Bundestagswahl werden an mehreren Stellen Informationen entschlüsselt. Daher muss gezeigt werden, dass diese Informationen nicht geheim sind, sondern sich direkt aus dem Ergebnis ableiten lassen.

Die erste Entschlüsselung findet statt, wenn wie in Abschnitt 6.4.2 beschrieben entschlüsselt wird, welche Parteien zu den ”großen Parteien” gehören. Die ”großen Parteien” sind als die Parteien definiert, die Minderheitsparteien sind, oder über der 5%-Hürde liegen oder mindestens drei Direktmandate erhalten. Es ist öffentlich bekannt, welche Parteien Minderheitsparteien sind und für diese werden die anderen beiden Bedingungen nicht überprüft. Daher wird hier keine geheime Information entschlüsselt. Die anderen beiden Bedingungen sind für Nicht-Minderheitsparteien äquivalent dazu, dass die Partei in der finalen Sitzverteilung mindestens drei Sitze erhält. Jede Partei, die mindestens drei Direktmandate gewinnt, erhält auch mindestens drei Sitze, da für die Direktmandate garantiert ist, dass sie ins Parlament einziehen. Außerdem erhält auch jede Partei über der 5%-Hürde mindestens drei Sitz in der Sainte-Laguë-Verteilung, die durch die zweite Oberverteilung errechnet wird. Dies lässt sich analog zu Theorem 12 der Wahl des norwegischen Parlaments zeigen. Bei der deutschen Bundestagswahl sind sowohl die Prozenthürde als auch die Mindestanzahl an Sitzen im Parlament höher als bei der norwegischen Wahl. Dadurch ist die Bedingung auch für drei Sitze erfüllt. Außerdem können im weiteren Verlauf keine Parteien, die nicht ”große Parteien” sind, Sitze erhalten. Daher ist die Menge der großen Parteien die Vereinigung aus Minderheitsparteien und den Parteien, die in der finalen Sitzverteilung mindestens drei Sitze erhalten. So werden in diesem Schritt keine Informationen entschlüsselt, die sich nicht aus dem Ergebnis oder öffentlichem Wissen ableiten lassen.

Eine weitere Entschlüsselung findet in der Methode `testSeats` statt, die in Algorithmus 6.6 aufgerufen wird. Hierbei wird für jede Partei und für eine beliebige Anzahl an Sitzen getestet, ob die Partei mindestens diese Anzahl an Sitzen erhält. Diese Information ist öffentlich bekannt, da im Ergebnis, das heißt in der Sitzverteilung im Parlament, auch die Information enthalten ist, wie viele Sitze eine Partei erhält.

Am Ende des übernächsten Schrittes, der die verbleibenden Überhänge auf die Bundesländer zuordnet, wird entschlüsselt, wie viele Sitze die Parteien in den einzelnen Bundesländern erhalten. Analog zur vorherigen Begründung muss auch diese Information nicht geheim bleiben. Da jeder Abgeordnete über die Landesliste beziehungsweise das Direktmandat eindeutig einem Bundesland zugeordnet ist, kann aus dem Ergebnis die Anzahl an Sitzen pro Partei pro Bundesland ermittelt werden.

Die letzte Entschlüsselung findet in Abschnitt "Feststellung des Ergebnisses" statt. Hier wird für jeden Kandidaten entschlüsselt, ob er als Abgeordneter in den Bundestag einzieht oder nicht. Dies entspricht genau dem Ergebnis der Wahl, welches nach Definition entschlüsselt werden darf. Damit wurde für alle Entschlüsselungen gezeigt, dass neben dem Ergebnis keine weiteren Informationen entschlüsselt werden, sodass alle Voraussetzungen erfüllt sind. ■

6.4.11 Benchmarks

Die Umsetzung der Bundestagswahl mit Ordinos wurde anhand der Daten der letzten Bundestagswahl [46] gebenchmarkt. Die Gesamtlaufzeit dieser Auswertung beträgt 38 Stunden und in Abbildung 6.10 ist dargestellt, welchen Anteil die einzelnen Schritte an der Gesamtlaufzeit haben. Im Gegensatz zu den vorherigen Umsetzungen ist es bei der Bundestagswahl schwierig einzuschätzen, wie sich die Gesamtlaufzeit bei zukünftigen Wahlen ändern könnte. Dies liegt daran, dass die Laufzeit von vielen Parametern abhängig ist wie beispielsweise der Gesamtanzahl an Sitzen, der Anzahl an großen Parteien und den Anteil an Wählern, die bei der Erst- und Zweitstimme unterschiedliche Parteien wählen. Daher wird nur für jeden Teilschritt eingeschätzt, wie sich die Laufzeit dieses Schrittes ändern könnte.

Die Berechnung der Direktmandate macht mit etwa einem Drittel einen großen Teil der Laufzeit aus. Da es 299 Wahlkreise gibt, ist nicht zu erwarten, dass die Gesamtzeit sich bei zukünftigen Wahlen merklich ändert, denn wie bei der Wahl des House of Commons werden sich die Änderungen der Kandidatenanzahlen in verschiedenen Wahlkreisen vermutlich ausgleichen. Die lange Laufzeit kommt hauptsächlich durch die große Anzahl an Wahlkreisen zustande, denn ein Wahlkreis benötigt im Durchschnitt $10,3$ Minuten Zeit für die Auswertung. Daher kann die Laufzeit durch mehr parallele Prozesse deutlich reduziert werden.

Die erste Oberverteilung und die Bestimmung der großen Parteien benötigen unter 10 Minuten Zeit. Der Grund hierfür ist, dass die erste Oberverteilung unverschlüsselt stattfinden kann und für die Bestimmung der großen Parteien pro Partei nur ein Vergleich mit kleinen Bitgrößen erfolgen muss.

Mit knapp über zwei Stunden ist auch die erste Unterverteilung recht schnell. Dies liegt daran, dass zwischen den Bundesländern parallelisiert werden kann und die Auswertungszeit somit durch das Bundesland mit den meisten Sitzen beschränkt ist. Dies ist konkret bei der letzten Bundestagswahl Nordrhein-Westfalen mit 127 Sitzen. Für diesen Schritt wird das schnelle Höchstzahlverfahren, das

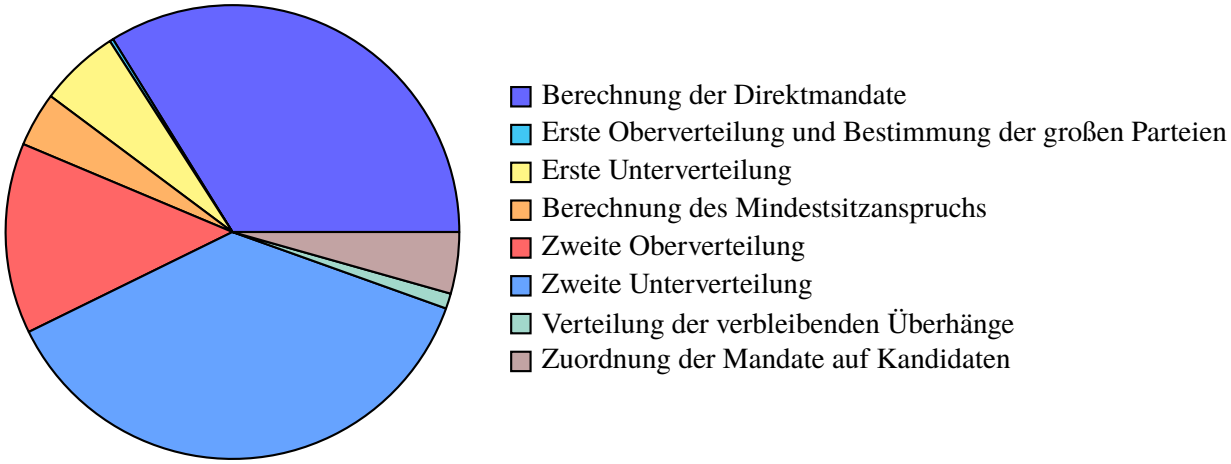


Abbildung 6.10: Benchmark der Auswertung der letzten Bundestagswahl mit Ordinos. Insgesamt benötigt die Auswertung 38 Stunden. Das Diagramm zeigt den Anteil der einzelnen Teilschritte. Die meiste Laufzeit kommt durch die Berechnung der Direktmandate und die zweite Unterverteilung zustande.

auf der Grundverteilung basiert, aus Abschnitt 6.1.3 verwendet. Das Höchstzahlverfahren könnte bei entsprechender Hardware für jedes Bundesland ebenfalls parallelisiert werden, sodass eine minimale Laufzeit von 52 Minuten möglich ist.

Die Berechnung des Mindestsitzanspruchs und die zweite Oberverteilung sind hauptsächlich davon abhängig, wie viele große Parteien vorliegen. Nach der zweiten Oberverteilung wird entschlüsselt, wie viele Sitze das neue Parlament hat. Daher sind die meisten folgenden Schritte abhängig von dieser Sitzzahl. Die Gesamtsitzzahl ist umso höher, je mehr Ausgleichsmandate vergeben werden müssen und die Ausgleichsmandate wiederum sind abhängig von der Anzahl der Überhangmandate. Daher werden umso weniger Sitze vergeben, je ähnlicher die Verteilung von Erst- und Zweitstimmen sind. Wenn also viele Wähler ihre beiden Stimmen der gleichen Partei geben, ist die Gesamtsitzzahl kleiner und die Auswertungszeit für die nächsten Schritte kürzer.

Die zweite Unterverteilung macht mit 37% einen beachtlichen Teil der Laufzeit aus. Sie wird für die einzelnen Parteien parallel berechnet und durch die Partei mit den meisten Sitzen beschränkt. Bei der letzten Bundestagswahl war dies die SPD mit 206 Sitzen [46]. Die Auswertung nach dem Sainte-Laguë-Verfahren benötigt deshalb so lang, da auf 16 Bundesländer anstatt wie bei der ersten Unterverteilung auf nur sechs Parteien verteilt werden muss. Da wie bei der ersten Unterverteilung das Höchstzahlverfahren parallelisiert werden kann, ist es möglich, die Laufzeit von aktuell 14 Stunden auf $6,5$ Stunden zu reduzieren. Zusätzlich kann die Laufzeit für die zweite Unterverteilung trivialerweise dadurch reduziert werden, dass mehr Parteien wie die CSU nur in einem Bundesland antreten. Dadurch entfällt für diese Parteien die Verteilung der Sitze auf die Bundesländer.

Die Verteilung der verbleibenden Überhänge ist wiederum abhängig von der Anzahl an Parteien und der letzte Schritt, die Zuordnung der Mandate auf die Kandidaten, ist abhängig von der Anzahl der Sitze, die eine Partei für ein Bundesland erhält. Diese Anzahl ist umso kleiner, je weniger Sitze insgesamt vergeben werden. Wie oben erklärt, ist dies hauptsächlich davon abhängig, wie viele Wähler beide Stimmen der gleichen Partei geben.

7 Zusammenfassung und Ausblick

In dieser Bachelorarbeit wurden Evaluierungsprotokolle für mehrere Parlamentswahlen aus der Praxis entwickelt und dem E-Voting System Ordinos hinzugefügt. Dabei wurde zu Beginn eine Literaturrecherche von Wahlen aus der Praxis durchgeführt, um geeignete Wahlen auszuwählen, die im Zuge dieser Arbeit umgesetzt werden. Im Gegensatz zu vorherigen Arbeiten lag der Fokus nicht darauf, generische Wahlschemata zu beschreiben. Stattdessen ist das Ziel, Wahlen aus der Praxis genau zu untersuchen und dabei Feinheiten, Sonderregelungen und die Vermischung verschiedener Wahlsysteme zu behandeln. Anschließend wurden einige allgemeine Änderungen für Wahlen aus der Praxis durchgeführt. Das sind vor allem das Tie-Breaking und das Wählen in Wahlkreisen, die in dieser Arbeit erstmals in Ordinos betrachtet wurden.

Daraufhin wurden im Hauptteil der Arbeit die drei Wahlverfahren des britischen House of Commons, des norwegischen Parlaments und des Deutschen Bundestags sowie das Sitzzuteilungsverfahren nach Sainte-Laguë in Ordinos umgesetzt. Dabei mussten häufig eigene Algorithmen entwickelt werden, da eine direkte Umsetzung der offiziellen Beschreibungen der Wahlen in Ordinos zu sehr langen Laufzeiten führt. Der Grund hierfür sind unterschiedliche Ziele der Optimierung: Die offiziellen Beschreibungen der Wahlen sind auf gute Lesbarkeit beziehungsweise gute Verständlichkeit der Verfahren optimiert. Ein Beispiel für die Lesbarkeit ist in der offiziellen Auswertung der Bundestagswahl. Hier wurde das Divisorverfahren angewandt und wenn ein Intervall an gültigen Divisoren vorliegt, wird immer der Divisor gewählt, der eine möglichst ”runde” Zahl ist [46]. Für die mathematische Auswertung bringt dies keinen Vorteil, doch die Lesbarkeit wird verbessert. Ein weiteres Beispiel ist die Verteilung der Ausgleichssitze in der norwegischen Wahl. Hier werden die Parteien, die mehr Bezirkssitze als Sitze nach der Sainte-Laguë-Verteilung erhalten, schrittweise entfernt. Dies trägt zur Verständlichkeit bei, da nur genau die Partei aus der Sainte-Laguë-Menge entfernt wird, für die intuitiv klar ist, dass sie nicht an der Sainte-Laguë-Verteilung teilnehmen kann. In der vorgestellten Umsetzung mit Ordinos hingegen werden die Parteien nach einem Quotienten aus Stimmen und Bezirkssitzen sortiert, doch die intuitive Bedeutung dieses Quotienten ist schwieriger zu erkennen.

Im Gegensatz dazu wird bei der Umsetzung in Ordinos in Bezug auf die Laufzeit optimiert. Die einzelnen Operationen benötigen durch die Berechnungen im homomorphen Paillier-Verschlüsselungsschema verhältnismäßig viel Zeit, sodass die Auswertung einer Wahl mindestens im Stundenbereich liegt. Daher sind Laufzeitoptimierungen deutlich wichtiger als beim klassischen Programmieren mit unverschlüsselten Zahlen. Außerdem ist bei der Umsetzung mit Ordinos zu beachten, dass die arithmetischen und logischen Operationen unterschiedlich gewichtet sind. So benötigen beispielsweise Vergleiche viel Laufzeit, und im Fall von Schleifen oder Verzweigungen müssen immer alle Pfade berechnet werden. All diese Anforderungen waren bei der Entwicklung der Algorithmen für die Evaluierungsprotokolle zu beachten.

Im weiteren Verlauf der Arbeit wurde durch Benchmarks gezeigt, dass die Laufzeit der Protokolle in einer angemessenen Zeit liegt. Die Protokolle wurden mit realen Werten der jeweils letzten Wahl gebenchmarkt und nach Möglichkeit eingeschätzt, wie sich die Laufzeiten bei zukünftigen Wahlen verändern könnten. Außerdem wurde gezeigt, dass alle Sicherheitseigenschaften erfüllt sind, sodass die entwickelten Evaluierungsfunktionen sicher in das Grundgerüst von Ordinos eingefügt werden können. Damit wurde das Ziel erreicht, Ordinos um Wahlverfahren aus der Praxis zu erweitern.

In zukünftigen Arbeiten könnte man diesen Weg in die Praxis weiterschreiten. Zwar gibt es in Ordinos jetzt Evaluierungsfunktionen, die solche Wahlen auswerten können, doch für die Anwendung bei großen politischen Wahlen ist das gesamte E-Voting System noch nicht bereit. Dafür müssten weitere Sicherheitseigenschaften wie beispielsweise Availability garantiert werden und es müssten auch weitere Arten von Angreifern beachtet werden, die beispielsweise kein Interesse haben, das Wahlverhalten des Wählers unter Beobachtung zu ermitteln, sondern die den Wahlprozess stören wollen. Ein weiterer Ansatz für zukünftige Arbeiten ist, die arithmetischen und logischen Grundoperationen der Trustees so anzupassen, dass tatsächliche Parallelisierung möglich ist. In Quelle [31] wurde bereits gezeigt, dass die Trustees auch über ein Netzwerk kommunizieren können. Dadurch könnte erreicht werden, dass mehr parallele Prozesse zur Verfügung stehen, um die Laufzeit der Auswertung weiter zu reduzieren. Außerdem kann Ordinos auch um weitere unterschiedliche Wahlverfahren ergänzt werden. Insbesondere für die Aggregation der Stimmzettel könnten sich neue Verfahren überlegt werden, die auf Stimmzettel mit Rankings spezialisiert sind. Mit dessen Hilfe könnte die Umsetzung von Verfahren wie der Wahl des irischen Parlaments ermöglicht werden.

Literaturverzeichnis

[1] URL: https://www.evoting.biz/ (zitiert auf S. 13).

[2] URL: https://www.polyas.de/online-wahlen (zitiert auf S. 13).

[3] URL: https://www.demokratiezentrum.org/bildung/ressourcen/lexikon/e-voting/ (zitiert auf S. 13).

[4] *Act relating to parliamentary and local government elections (Election Act)*. 2022. URL: https://lovdata.no/dokument/NLE/lov/2002-06-28-57/KAPITTEL_1 (zitiert auf S. 35, 37, 48).

[5] B. Adida. „Helios: Web-based Open-Audit Voting." In: *USENIX security symposium*. Bd. 17. 2008, S. 335–348 (zitiert auf S. 25).

[6] M. L. Balinski, H. P. Young. „The Webster method of apportionment". In: *Proceedings of the National Academy of Sciences* 77.1 (1980), S. 1–4 (zitiert auf S. 65).

[7] B. Beckmann, G. Trenkler, F. Pukelsheim. „Das Landtagswahlsystem in Nordrhein-Westfalen". Diss. Diploma thesis, Universität Dortmund, 2006 (zitiert auf S. 32–34).

[8] *Bundestagswahlen - Kandidatinnen und Kandidaten*. 2021. URL: https://www.bpb.de/kurz-knapp/zahlen-und-fakten/bundestagswahlen/339917/kandidatinnen-und-kandidaten/ (zitiert auf S. 80).

[9] *Bundeswahlgesetz (BWG)*. 2021. URL: https://www.bundeswahlleiter.de/dam/jcr/2596ba8d-34e4-4c9b-a731-a27f8fb0618f/bundeswahlgesetz.pdf (zitiert auf S. 38).

[10] V. Cortier, P. Gaudry, Q. Yang. „A toolbox for verifiable tally-hiding e-voting systems". In: (2021) (zitiert auf S. 25, 48).

[11] *Dáil Éireann - 33rd Dáil General Election*. 2020. URL: https://data.oireachtas.ie/ie/oireachtas/electoralProcess/electionResults/dail/2020/2020-05-01_33rd-dail-general-election-results_en.pdf (zitiert auf S. 30, 31).

[12] I. Damgård, M. Jurik, J. B. Nielsen. „A generalization of Paillier's public-key system with applications to electronic voting". In: *International Journal of Information Security* 9.6 (2010), S. 371–385 (zitiert auf S. 25).

[13] *Debian-Satzung*. 2016. URL: https://www.debian.org/devel/constitution (zitiert auf S. 27).

[14] A. Dopatka. „E-Voting in Deutschland? Zum Problem der Stimmabgabe über das Internet bei politischen Wahlen". Diss. Stiftung Universität Hildesheim, 2005 (zitiert auf S. 13).

[15] M. Fehndrich. *Das Divisorverfahren mit Standardrundunge*. 1999. URL: https://www.wahlrecht.de/verfahren/stlague.html (zitiert auf S. 31, 33).

[16] M. Fehndrich. *Frankreich (Assemblée nationale)*. 2002. URL: https://www.wahlrecht.de/ausland/franzoesisch.html (zitiert auf S. 29).

[17] M. Fehndrich, M. Cantow. *Präsidentschaftswahl 2017 in Frankreich*. 2002. URL: https://www.wahlrecht.de/ausland/frankreich.html (zitiert auf S. 29).

[18] M. Fehndrich, K. Karpinsky. *Großbritannien (House of Commons)*. 2001. URL: https://www.wahlrecht.de/ausland/grossbritannien-parlament.html (zitiert auf S. 28, 45, 70).

[19] M. Fehndrich, W. Zicht, K. Karpinsky, M. Cantow. *Die Wahl des Präsidenten der USA*. 2001. URL: https://www.wahlrecht.de/ausland/us-praesident.html (zitiert auf S. 28).

[20] *French Republic - Election for Assemblée Nationale (National Assembly of France)*. 2017. URL: https://www.electionguide.org/elections/id/2570/ (zitiert auf S. 29).

[21] *General Election 2019: results and analysis*. 2019. URL: https://researchbriefings.files.parliament.uk/documents/CBP-8749/CBP-8749.pdf (zitiert auf S. 28, 71).

[22] G. Genssler. „Das d'Hondtsche Verfahren und andere Sitzverteilungsverfahren aus mathematischer und verfassungsrechtlicher Sicht". Diss. Schnell-Druck-Diana, 1984 (zitiert auf S. 61, 63, 65).

[23] *Gesetz über das Bundesverfassungsgericht, § 7*. URL: https://www.gesetze-im-internet.de/bverfgg/__7.html (zitiert auf S. 27).

[24] F. Hertel. „Erweiterung des E-Voting-Systems Ordinos um weitere Wahlverfahren". B.S. thesis. 2020 (zitiert auf S. 13, 27).

[25] F. Hertel, N. Huber, J. Kittelberger, R. Küsters, J. Liedtke, D. Rausch. „Extending the Tally-Hiding Ordinos System: Implementations for Borda, Hare-Niemeyer, Condorcet, and Instant-Runoff Voting". In: *Cryptology ePrint Archive* (2021) (zitiert auf S. 3, 15, 18–20, 25, 31, 47, 50, 53, 60–62).

[26] A. Hylland. *ELECTIONS IN NORWAY - Notes on the electoral system*. 2010. URL: https://www.sv.uio.no/econ/english/people/aca/aanundh/upubliserte-artikler-og-notater/Norwegian%20elections%202010%5B1%5D.pdf (zitiert auf S. 34, 35, 74, 78, 80).

[27] *Innst. 1 S (2021-2022) - Innstilling til Stortinget fra fullmaktskomiteen*. 2021. URL: https://www.stortinget.no/globalassets/pdf/innstillinger/stortinget/2021-2022/inns-202122-001s.pdf (zitiert auf S. 35, 50, 72, 74).

[28] W. Jamroga, P. B. Roenne, P. Y. Ryan, P. B. Stark. „Risk-limiting tallies". In: *International Joint Conference on Electronic Voting*. Springer. 2019, S. 183–199 (zitiert auf S. 25).

[29] J. Kelly. *When the flip of a coin wins an election*. 2016. URL: https://www.bbc.com/news/magazine-35473068 (zitiert auf S. 28).

[30] K. Kopfermann. *Mathematische Aspekte der Wahlverfahren: Mandatsverteilung bei Abstimmungen*. BI Wissenschaftsverlag, 1991 (zitiert auf S. 75).

[31] R. Küsters, J. Liedtke, J. Müller, D. Rausch, A. Vogt. „Ordinos: A verifiable tally-hiding e-voting system". In: *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE. 2020, S. 216–235 (zitiert auf S. 3, 13, 15–20, 25, 44, 47, 60, 68, 90).

[32] H. Lipmaa, T. Toft. „Secure equality and greater-than tests with sublinear online complexity". In: *International Colloquium on Automata, Languages, and Programming*. Springer. 2013, S. 645–656 (zitiert auf S. 21).

[33] H. Lipmaa, T. Toft. „Secure equality and greater-than tests with sublinear online complexity". In: *International Colloquium on Automata, Languages, and Programming*. Springer. 2013, S. 645–656 (zitiert auf S. 25).

[34] *Ordinos Code Reprository*. URL: https://github.com/JulianLiedtke/ordinos (zitiert auf S. 58, 70).

[35] *Parlamentswahl in Norwegen 2021*. 2021. URL: https://www.bpb.de/kurz-knapp/hintergrund-aktuell/340084/parlamentswahl-in-norwegen-2021/ (zitiert auf S. 34).

[36] *Proportional representation*. 2020. URL: https://www.citizensinformation.ie/en/government_in_ireland/elections_and_referenda/voting/proportional_representation.html (zitiert auf S. 30, 31).

[37] *Regensburg/Gründung/Geschäftsordnung*. URL: https://wiki.piratenpartei.de/BY:Regensburg/Gr%C3%BCndung/Gesch%C3%A4ftsordnung#Ausz.C3.A4hlung (zitiert auf S. 27).

[38] M. Reiners. *E-Voting in Estland: Vorbild für Deutschland?* 2017. URL: https://www.bpb.de/shop/zeitschriften/apuz/255967/e-voting-in-estland-vorbild-fuer-deutschland/ (zitiert auf S. 13).

[39] *Resolving Tied Elections for Legislative Offices*. 2021. URL: https://www.ncsl.org/research/elections-and-campaigns/resolving-tied-elections.aspx (zitiert auf S. 28, 46).

[40] *Returning officer and electoral registration officer*. URL: https://www.rbwm.gov.uk/home/council-and-democracy/elections-and-voting/returning-officer-and-electoral-registration-officer (zitiert auf S. 28).

[41] *Satzung der Universität Stuttgart zur Durchführung der Gremienwahlen (Wahlordnung – WahlO)*. 2019. URL: https://www.beschaeftigte.uni-stuttgart.de/uni-services/recht/dokumente/Wahlordnung_Ausfertigung_15_02_2019.pdf (zitiert auf S. 27).

[42] B. Schoenmakers, M. Veeningen. „Universally verifiable multiparty computation from threshold homomorphic cryptosystems“. In: *International Conference on Applied Cryptography and Network Security*. Springer. 2015, S. 3–22 (zitiert auf S. 20).

[43] B. Terelius, D. Wikström. „Proofs of restricted shuffles“. In: *International Conference on Cryptology in Africa*. Springer. 2010, S. 100–113 (zitiert auf S. 47).

[44] *Verfassung des Landes Baden-Württemberg*. 1953. URL: https://www.landesrecht-bw.de/jportal/?quelle=jlink&query=Verf+BW&psml=bsbawueprod.psml&max=true&aiz=true#jlr-VerfBWV22Art60 (zitiert auf S. 29).

[45] *Volksmehr und Ständemehr*. URL: https://www.ch.ch/de/abstimmungen-und-wahlen/abstimmungen/volksmehr-und-standemehr/ (zitiert auf S. 29).

[46] *Wahl zum 20. Deutschen Bundestag am 26. September 2021 – Heft 3*. 2021. URL: https://bundeswahlleiter.de/dam/jcr/cbceef6c-19ec-437b-a894-3611be8ae886/btw21_heft3.pdf (zitiert auf S. 31–33, 38, 40–42, 45, 46, 48, 50, 52, 80, 83, 86–89).

[47] W. Zicht. *Wahlsysteme in den EU-Mitgliedstaaten*. 2004. URL: https://www.wahlrecht.de/ausland/europa.htm (zitiert auf S. 30).

Alle URLs wurden zuletzt am 12. 04. 2022 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift