FH CAMPUS WIEN
UNIVERSITY OF APPLIED SCIENCES

Ein elektronisches Wahlprotokoll mit Hilfe von Blindsignaturen auf Gruppenelementen

An Electronic Voting Protocol with Blind Signatures on Group Elements

Masterarbeit

Zur Erlangung des akademischen Grades

Master of Science in Engineering

der Fachhochschule Campus Wien
Masterstudiengang: IT-Security

Vorgelegt von:
Andrea Kapsch

Personenkennzeichen:
1810537025

Erstbetreuer / Erstbegutachter:
Dr. Ulrich Haböck

Eingereicht am:
13.05.2020

Erklärung:

Ich erkläre, dass die vorliegende Masterarbeit von mir selbst verfasst wurde und ich keine anderen als die angeführten Behelfe verwendet bzw. mich auch sonst keiner unerlaubter Hilfe bedient habe.

Ich versichere, dass ich diese Masterarbeit bisher weder im In- noch im Ausland (einer Beurteilerin/einem Beurteiler zur Begutachtung) in irgendeiner Form als Prüfungsarbeit vorgelegt habe.

Weiters versichere ich, dass die von mir eingereichten Exemplare (ausgedruckt und elektronisch) identisch sind.

Datum: Unterschrift:

Kurzfassung

Das Thema der elektronischen Wahlen ist in der heutigen Zeit sehr präsent. Aufgrund der wachsenden Ansprüche in der digitalen Welt ist Sicherheit auch im Bereich der elektronischen Wahlen von hoher Wichtigkeit. Es ist notwendig einen gewissen Sicherheits-Standard bei der Einführung eines neuen Wahlsystems zu bieten. In dieser Arbeit werden die wichtigsten existierenden elektronischen Wahlsysteme vorgestellt. Diese werden im Anschluss anhand von festgelegten Faktoren verglichen. Bei dem Vergleich zeigt sich, dass die Komplexität mit der Anzahl der zu erfüllenden Bedingungen ansteigt. Es wird außerdem aufbauend auf einer restriktiven, verblindeten Version der AGHO Signatur auf Gruppenelemente, die es dem User / der Userin erlaubt eine zu signierende Nachricht vor dem Unterzeichner / der Unterzeichnerin zu verbergen, ein elektronisches Wahlprotokoll vorgestellt. Dieses Protokoll wurde als kryptographische Bibliothek, bestehend aus drei Komponenten implementiert: die Pairing-basierte AGHO Signatur, die ElGamal Verschlüsselung auf elliptischen Kurven und die benötigten nicht-interaktiven Zero-Knowledge Proofs. Zur Überprüfung der Anwendbarkeit dieser Bibliothek wurde eine prototypische Implementierung eines Client-Server Modells durchgeführt. Diese wird auf dieselben Sicherheits-Bedingungen, wie die zuvor vorgestellten Systeme getestet. Es zeigt sich, dass 9 von 11 der Bedingungen bereits erfüllt sind oder durch kleine Erweiterungen schnell erreicht werden können. Zum Schluss wird auf Erweiterungs-Möglichkeiten der erstellten Implementierung eingegangen.

Abstract

The topic of electronic voting systems is very present these days. Due to the growing requirements in our digital world, security regarding to electronic voting protocols is also of great importance. It is necessary to provide a certain security level when implementing and launching a new voting system. In this work the most important existing electronic voting systems are presented and described. These are compared based on specific security-related factors. The comparison shows that the complexity of the system increases with the number of conditions to be met. Based on a restrictive blind version of the AGHO signature on group elements, which allows the user to hide a message, that should be issued, from the signer, a new version and a new electronic voting protocol is presented. This protocol is implemented as a cryptographic library consisting of three components: the pairing-based AGHO signature, the ElGamal encryption over elliptic curves and the needed non-interactive Schnorr type zero-knowledge proofs. A prototype client-server implementation has been made to check the applicability of the cryptographic library. In this work this implementation is reviewed for the same security conditions as the previously presented systems. It can be seen that 9 out of 11 of the conditions have been met or can be achieved quickly with a few modifications or extensions. Finally, options for expanding the resulting implementation are discussed.

Abkürzungsverzeichnis

A	Auszählbehörde
AS	Authentifizierungs Server
AGHO	Abe, Groth, Haralambiev und Ohkubo
BB	Bulletin Board
BPS	Ballot Preparation System
C	Client
DDOS	Distributed Denial of Service
DKG	Distributed Key Generator
DVBM	Digital Vote by Mail Service
EC	Elliptic Curve
ECC	Elliptic Curve Cryptography
EV	Encrypted Vote
ggT	größter gemeinsamer Teiler
NIZK	Nicht-interaktiver Zero-Knowledge Proof
P	BeweiserIn (ProverIn)
PKC	Public Key Cryptography
PKI	Public Key Infrastructure
PRG	Pseudo Random Generator
RSA	Rivest, Shamir und Adleman
V	VerifiziererIn
VAV	Vote, Anti-Vote, Vote
VSD	Voter Supporting Devices
W	Wahlamt/Wahlbehörde
WBB	Web Bulletin Board
ZKP	Zero-Knowledge Proof

Schlüsselbegriffe

AGHO Signaturverfahren
Bilineare Gruppen
Blindsignatur
Elektronische Wahlen
Electronic Voting Systems
ElGamal Verschlüsselung
Elliptic Curve Cryptography
Pairings
Remote Wahlsysteme

Inhaltsverzeichnis

1 Einführung

Alle Prozesse, Tätigkeiten und Aufgaben müssen in der heutigen Zeit mit einem hohen Maß an Qualität und Sicherheit schnell zu guten und aussagekräftigen Ergebnissen führen. Ähnliche und einheitliche Schritte werden so weit wie möglich automatisiert, um Abläufe einfacher, aber auch effizienter gestalten und abwickeln zu können.
Da wir in einer sehr schnelllebigen Gesellschaft leben, sollen langwierige Amtswege, wie wählen gehen, effizienter gestaltet werden können. Es gibt bereits viele verschiedene Ansätze, wie und in welchem Ausmaß der Wahlprozess automatisiert werden kann und auch schon einige Systeme, welche diese Ansätze umsetzen. Angefangen mit der Überprüfung der Identität vor der Wahlkabine bis hin zur vollautomatisierten Wahl über das Internet von zu Hause aus, gibt es hierbei eine Vielfalt an Möglichkeiten [BBH+17]. Bei allen Ansätzen ist es wichtig die Wahl, sowohl für die Wähler und Wählerinnen aber auch für alle anderen Beteiligten, sicher und nicht manipulierbar zu gestalten. Dies bedeutet, dass sowohl die bisher existierenden Sicherheitsmaßnahmen miteinbezogen, sowie auch neue Einschränkungen, welche durch die Automatisierung hinzukommen, berücksichtigt werden müssen. Der Aufwand, der für die Umsetzung dieser Maßnahmen notwendig ist, ist unter anderem auch ein Grund dafür, wieso heute viele der existierenden Systeme nicht mehr für demokratische Wahlen zum Einsatz kommen.
Um eine Sicherheitsstufe zu erreichen, welche dem Stand der Technik entspricht, müssen Richtlinien definiert werden, die genau beschreiben, welche Kriterien bei elektronischen Wahlsystemen erfüllt sein müssen, um diese auch in der Praxis anwendbar zu machen.
Grundsätzlich können elektronische Wahlen in zwei Hauptkategorien eingeteilt werden [BBH+17]:

- Systeme mit Wahllokalen (Pollsite Systems)
- Remote Systeme (Remote Systems)

Bei den Systemen mit Wahllokalen findet die Wahl, wie der Name schon sagt, immer noch in den Wahllokalen statt. Ein Beispiel hierfür wäre *Scantegrity* [CCC+08b, CEC+08], bei dem mit optischen Scan-Stimmzetteln gearbeitet wird oder auch das *ThreeBallot* System [Riv06], bei dem eine Stimme über drei Stimmzettel abgegeben wird. Zur zweiten Kategorie, welche jene ist, die für diese Masterarbeit interessant ist, zählen die Internet-Wahlen. Ein Beispiel für Remote Systeme wäre *Remotegrity* [ZCC+13]. Hierbei wird den Wählern und Wählerinnen der Stimmzettel mit einer Autorisierungskarte, welche Codes für das Durchführen der Wahl enthält, zugesendet. Somit kann die Wahl in diesem Fall von zu Hause aus durchgeführt werden [BBH+17]. Weitere Beispiele für Remote Systeme wären *Civitas* [CCM08] bei dem die WahlhelferInnen verschiedene Rollen annehmen und unterschiedliche Aufgabenbereiche haben und *sElect* [KSM+16], was als Plattform unabhängige Webapplikation implementiert wird.

In dieser Masterarbeit werden zuerst einige existierende elektronische Wahlsysteme vorgestellt. Es wird dabei auf die Funktionsweise und den Ablauf sowie deren Eigenschaften eingegangen. Anschließend werden die Systeme hinsichtlich ihrer Sicherheitseigenschaften, wie zum Beispiel Integrität, Nachweisbarkeit und Privatsphäre, miteinander verglichen. Dieser Vergleich dient dazu einen ersten Eindruck von der Erfüllbarkeit der einzelnen Sicherheitsei-

genschaften zu bekommen.
Anschließend werden die benötigten kryptographischen Grundlagen wiederholt, um eine geeignete Wissensbasis für die Erläuterung der AGHO Signatur [AGHO11] bzw. der restriktiven AGHO Blindsignatur [HK19] zu erreichen. Dieses Signaturverfahren ist die Grundlage für das später formulierte Remote Wahlsystem. In Kapitel 4 wird dieses System in Form einer kryptographischen Bibliothek vorgestellt und auf die Implementierung und deren Schwierigkeiten eingegangen. Außerdem wird die Implementierung auf die zuvor beschriebenen Sicherheitseigenschaften überprüft.
Im nächsten Schritt wird als praktische Anwendung für die entstandene Bibliothek eine prototypische Implementierung eines Client-Server Modells vorgestellt. Es wird darauf eingegangen, welches Potential sich durch diesen Prototyp entwickelt und welche Sicherheitseigenschaften durch die Implementierung zusätzlich erfüllt werden. Zum Abschluss werden die Ergebnisse zusammengefasst und es wird diskutiert, welche Erweiterungsmöglichkeiten sich durch diese Masterarbeit ergeben.

2 Hintergrund

2.1 Sicherheitsbegriffe für ein elektronisches Wahlsystem

Um Ereignisse, deren Sicherheit und Korrektheit von großer Bedeutung sind, automatisiert oder teilweise automatisiert abhalten zu können, ist es notwendig bestimmte Regeln aufzustellen. Wenn es um Wahlen geht sollten diese Regeln beschreiben, welche Voraussetzungen ein System bzw. ein Protokoll erfüllen muss, um die bisher gesetzten Vorgaben für das Abhalten einer Wahl einzuhalten. Zusätzlich müssen auch die durch die Automatisierung neu auftretenden Aspekte miteinbezogen werden. Hierfür können nach [SMH17] 11 Bedingungen definiert werden:

- **Verfügbarkeit (Availability)**
  Das System muss für die Dauer der Wahl verfügbar sein und die Nutzer und Nutzerinnen davor bewahren sich mit nicht vertrauenswürdigen Clients zu verbinden.
- **Anonymität (Anonymity and Election Secrecy)**
  Es darf nicht möglich sein, eine Verbindung zwischen einer Stimme und dem dazugehörigen Benutzer oder der dazugehörigen Benutzerin, ohne der Hilfe des Benutzers oder der Benutzerin selbst, herzustellen.
- **Wähler Nachweisbarkeit (Voter Verifiablility)**
  Der Wähler oder die Wählerin selbst kann feststellen, dass seine oder ihre Stimme bei der Wahl gezählt wurde.
- **Universelle Nachweisbarkeit (Universal Verifiability)**
  Nach der Auszählung muss es möglich sein das Ergebnis von einer beliebigen Person überprüfen zu lassen.
- **Integrität (Integrity)**
  Die Integrität der Wahl muss gegeben sein.
- **Wählbarkeit (Eligibility)**
  Nur wahlberechtigte Personen sollen fähig sein zu wählen und diese sollen auch nur einmal wählen können.
- **Keine Belegbarkeit (Receipt Freeness)**
  Es können aus der Wahl keinerlei Informationen gewonnen werden. Das heißt, dass der Wähler oder die Wählerin einer dritten Partei nicht beweisen kann, wie er oder sie gewählt hat.
- **Robustheit (Robustness)**
  Das System muss mit fehlerhaften Stimmen umgehen können (zum Beispiel, wenn ungültig gewählt wurde).
- **Fairness**
  Es dürfen keine Teilergebnisse veröffentlicht werden, bevor die Wahl nicht beendet ist.
- **Korrektheit (Correctness)**
  Die Wahlergebnisse müssen fachgerecht ausgezählt und korrekt veröffentlicht werden.
- **Zwang zur Wahl (Coercion)**

Es darf nicht möglich sein, dass eine Person eine andere zwingt eine gewisse Auswahl zu treffen. Das bedeutet auch, dass das Verkaufen von Zugangsdaten und Berechtigungen nicht ermöglicht werden darf.

Bis jetzt ist noch kein Protokoll bekannt, welches alle diese Bedingungen erfüllt, da sich einzelne Bedingungen in manchen Punkten widersprechen [BBH+17]. Vor allem die letzte Bedingung *Zwang zur Wahl* ist für Remote Systeme sehr schwer zu erfüllen. Es darf keine Möglichkeit geben, dass eine Person die Wahl für jemand anderen durchführen kann. Es muss aber auch gewährleistet werden, dass kein Beweis erbracht werden kann, welche Wahl getroffen wurde, um einem Erpresser oder einer Erpresserin keinen Nachweis über die korrekte Auswahl bringen zu können.

2.2 Überblick über existierende elektronische Wahlsysteme

In diesem Abschnitt werden einige elektronische Wahlsysteme vorgestellt. Diese Systeme konnten nach einer ausführlichen Literaturrecherche als die Bekanntesten identifiziert werden. Hierbei wurde darauf geachtet, dass Systeme vorgestellt werden, welche verschiedene Herangehensweisen für das Durchführen von elektronischen Wahlen bieten aber auch dem aktuellen Stand der Technik entsprechen. Elektronische Wahlsysteme können, wie bereits in Kapitel 1 erklärt, in Systeme mit Wahllokalen und Remote Systeme unterteilt werden.

2.2.1 Systeme mit Wahllokalen

Prêt à Voter

Der Ablauf dieses Wahlsystems kann in vier Schritte unterteilt werden[RBHS10]:

- Stimmzettel-Generierung
- Einsammeln der Stimmen
- Stimmenauszählung
- Überprüfen auf Korrektheit

Donald	
Barack	
Alice	
Crystal	
Edward	
	a6Gq21p

Abbildung 2.1: Prêt à Voter Stimmzettel nach [RBHS10]

Der Stimmzettel ist so aufgebaut, dass die KandidatInnen auf der linken Seite des Stimmzettels in pseudozufälliger Reihenfolge aufgelistet sind. Auf der rechten Seite sollen die WählerInnen ihre Stimmen abgeben. Außerdem ist auf der Seite der Stimmabgabe eine verschlüsselte Information enthalten, die es dem System erlaubt später wieder die Auflistungs-Reihenfolge der KandidatInnen auf dem jeweiligen Stimmzettel zu errechnen (siehe Abbildung 2.1). Das bedeutet, dass die Auswahl der WählerInnen für die Auszählung aus der Position des Kreuzes

bestimmt werden kann. Wichtig ist hierbei, dass die KandidatInnen-Reihenfolge so verschlüsselt ist, dass keine Partei alleine die Entschlüsselung durchführen kann [RBHS10, BBH+17]. Durch die zufällige Reihenfolge der KandidatInnen wird die WählerInnen-Privatsphäre gewährleistet, da die Seite mit der KandidatInnenliste vor dem Einscannen des Stimmzettels entfernt und vernichtet wird [RBHS10, BBH+17].
Beim *Einsammeln der Stimmen* werden die rechten Seiten der Stimmzettel eingelesen und zur Wahl-Datenbank gesendet. Außerdem erhalten die WählerInnen die rechte Seite des eigenen Stimmzettels als Quittung, welche zusätzlich von der Maschine markiert bzw. signiert wird. Die rechten Seiten aller Stimmzettel werden veröffentlicht, damit die WählerInnen überprüfen können, ob ihre Stimme korrekt aufgenommen wurde [RBHS10].
Bei der *Stimmenauszählung* werden die verschlüsselten Stimmen in entschlüsselte Stimmen transformiert. Um die Entschlüsselung so vorzunehmen, dass niemand Ende-zu-Ende Vergleiche durchführen kann, das kann zum Beispiel das Durchführen eines Vergleichs der Abgabereihenfolge mit der Auszählungs-Reihenfolge sein, um so eine Stimme einem Wähler oder einer Wählerin zuordnen zu können, wird der Prozess in drei Tasks geteilt: *mixen*, *entschlüsseln* und *auszählen* [RBHS10].
Beim *Mixen* werden die verschlüsselten Stimmen an verschiedene Mix-Servers übergeben, welche die Stimmen dann ein oder mehrere Male durchmischen. Die Stimmen werden jeweils nach dem Mischen, dem Entschlüsseln und dem Auszählen veröffentlicht.[RBHS10].
Beim *Überprüfen auf Korrektheit* werden die im vorherigen Schritt durchgeführten Phasen auf ihre Richtigkeit überprüft. Dies kann zum Beispiel ein Zero-Knowledge Proof oder das Veröffentlichen aller entschlüsselten Stimmen sein.[RBHS10].

Scantegrity

Bei diesem elektronischen Wahlsystem wird mit optischen Scan-Stimmzetteln gearbeitet (siehe Abbildung 2.2). Dabei besteht ein Stimmzettel aus zwei Teilen: dem Wahl-Teil (oben) und dem Quittungs-Teil (unten). Auf jedem dieser zwei Teile ist eine Identifikationsnummer zu finden, welche für jeden Stimmzettel eindeutig ist [CCC+08a].

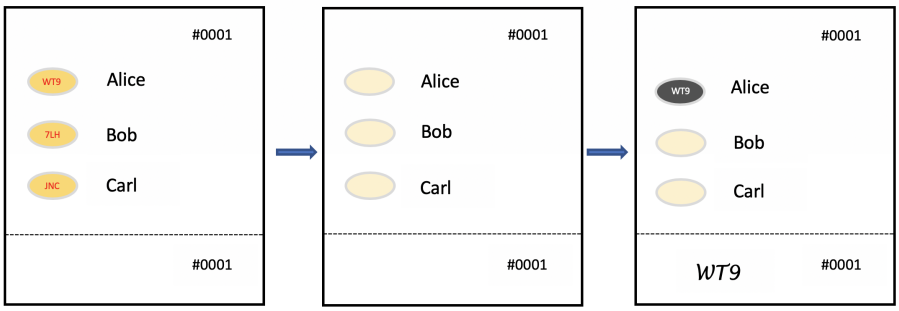

Abbildung 2.2: Der Scantegrity Stimmzettel nach [CCC+08a]

Auf dem Wahl-Teil des Stimmzettels befindet sich eine KandidatInnenliste mit einem Auswahlfeld neben jedem Namen. In jedem Auswahlfeld befindet sich eine zufällige Folge von alphanumerischen Zeichen, der *Bestätigungscode*. Diese Codes sind mit unsichtbarer Tinte in die Auswahlfenster eingedruckt und zu Beginn der Wahl noch nicht sichtbar (mittlere Abbildung in 2.2). Außerdem müssen diese Codes aller Stimmzettel einer Wahl und auch auf jedem einzelnen Stimmzettel eindeutig und auch pseudozufällig generiert sein. Die Bestätigungscodes sind den WählerInnen nicht bekannt, solange sie noch keine Auswahl getroffen haben und dadurch der Code freigelegt wird [CCC+08a].

Wenn ein Wähler oder eine Wählerin eine Stimme abgeben möchte so fragt dieser oder diese nach einem oder zwei Stimmzetteln und einem *Dekodierungsstift*. Ein Wähler oder eine Wählerin kann sich dafür entscheiden zwei Stimmzettel zu erhalten, damit diese/r mit einem der Stimmzettel die Stimme abgeben kann und mit dem anderen das System auditieren kann [CCC+08a].
Wenn ein Wähler oder eine Wählerin eine Auswahl treffen möchte, so muss dieser oder diese mit dem Dekodierungsstift das Auswahlfenster ausmalen. Dabei wird der Bestätigungscode sichtbar und das Fenster bekommt einen dunklen Hintergrund, was dem System dabei hilft die Auswahl zu erkennen [BBH+17, CCC+08a].
Der untere Teil des Stimmzettels, der Quittungs-Teil, beinhaltet genug Platz, damit sich WählerInnen den sichtbar gewordenen Bestätigungscode notieren können. Dieser Teil kann dann vom Stimmzettel abgetrennt und als Quittung aufbewahrt werden [CCC+08a].
Die Stimme kann schlussendlich abgegeben werden, indem der Stimmzettel an den optischen Scanner übergeben wird, welcher diesen auf Korrektheit (richtige Anzahl an ausgewählten KandidatInnen) überprüft. Sollte der Stimmzettel nicht gültig sein, kann ein neuer für eine erneute Stimmabgabe bei den WahlhelferInnen verlangt werden [CCC+08a].
Bei Gültigkeit markiert ein Wahlhelfer oder eine Wahlhelferin die Quittung, um zu erkennen, dass diese Quittung zu einem korrekt abgegebenen Stimmzettel gehört [CCC+08a].
Nach der Wahl wird eine Liste aller WählerInnen, die eine Stimme abgegeben haben, eine Liste der IDs aller Stimmzettel sowie eine Liste aller sichtbar gewordenen Bestätigungscodes veröffentlicht. Zusammen mit diesen Daten wird auch das Ergebnis bekannt gegeben [CCC+08a].

STAR-Vote

Dieses System ist ein Touchscreen-System, welches einen menschenlesbaren Stimmzettel druckt. Im Gegensatz zu den davor vorgestellten Systemen läuft diesmal der Auswahlprozess elektronisch ab. Dass die Wahlmaschinen miteinander vernetzt sind, vereinfacht den Datenaustausch während und nach der Wahl [BBB+13, BBH+17].

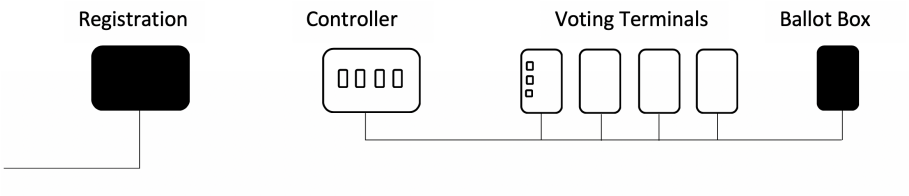

Abbildung 2.3: Die Wahlstationen von STAR-Vote aus der Sicht der WählerInnen nach [BBB+13]

Die WählerInnen müssen im Wahlablauf bei STAR-Vote die folgenden Stationen besuchen (siehe Abbildung 2.3) [BBB+13]:

Registrierung Mit einem Wahlhelfer muss der Check-In durchgeführt werden. Das Registrierungssystem selbst hat Zugriff auf eine zentrale Datenbank, um eine Veränderung im Status der einzelnen WählerInnen zu erkennen und somit das Doppelt-Wählen zu verhindern. Anschließend wird ein Sticker ausgedruckt, der den Namen und den Bezirk des Wählers oder der Wählerin enthält. Der Bezirk ist auch mit einem 1-D Barcode ausgestattet. Dieser Sticker kommt in ein Wahlbuch, welches der Wähler oder die Wählerin unterschreibt (als Backup zur Online-Datenbank).
Auch WählerInnen, welche nicht in der Wahl-Datenbank aufscheinen, haben die Chan-

ce zu wählen. Sie bekommen einen gewissen Präfix zu ihrem Bezirkscode und müssen an einer extra Maschine wählen. Wenn sie nachträglich vom System verifiziert wurden, kann so ihre Stimme trotzdem gezählt werden.

Controller Dieser scannt den Barcode auf dem Sticker, um den Bezirk der WählerInnen zu identifizieren. Der Controller druckt für den Wähler bzw. die Wählerin einen für das Wahllokal eindeutigen Code, mit dem der Wähler oder die Wählerin zu einem beliebigen Wahlterminal gehen kann und nach Eingabe des Codes den Stimmzettel erhält. Hierbei wird keine Verbindung zwischen dem Code und den WählerInnen gespeichert um die Anonymität zu gewährleisten.

Voting Terminals Die WählerInnen können auf den Terminals über die graphische Benutzeroberfläche ihre Auswahl treffen. Bevor die Auswahl abgeschlossen ist und der Stimmzettel gedruckt wird, gibt es die Möglichkeit den entstehenden Stimmzettel noch einmal auf dem Display zu sehen, um vom Wähler oder der Wählerin bestätigen zu lassen, dass der Stimmzettel korrekt ist.

Ist die Wahl von dem Wähler oder der Wählerin abgeschlossen, so wird vom System ein Commitment auf die Stimme erstellt und der menschenlesbare Stimmzettel, sowie eine Quittung (ein Hash über die Stimme) ausgedruckt. Der gedruckte Stimmzettel kann in eine Wahlurne (*Ballot Box*) geworfen werden [BBH$^+$17, BBB$^+$13].
Sollte ein Wähler eine Stimme zurückziehen wollen (z.B. aufgrund eines Fehlers oder um die Korrektheit des Systems zu überprüfen), so kann sich dieser an eine Mitarbeiterin wenden. Diese scannt den Code auf dem gedruckten Stimmzettel ein, um dem System damit zu signalisieren, dass dieser Stimmzettel zurückgezogen wird und somit nicht in die Auszählung eingeschlossen werden soll. Die passende Stimme muss daraufhin entschlüsselt und veröffentlicht werden, so als wäre die Wahl schon vorüber. Wenn das System keinen Beweis darüber erstellen kann, dass die entschlüsselte Stimme zu dem abgegebenen Klartext gehört, dann wurde das System als unehrlich enttarnt. Sollte diese Verifikation nicht gewünscht sein, kann der Auswahlprozess auch einfach von Neuem gestartet werden [BBH$^+$17, BBB$^+$13].
Nach der Wahl sendet das System die Schwellwert-verschlüsselten Stimmzettel und die verworfenen Stimmzettel an ein *Web Bulletin Board* (WBB), wo von den WählerInnen überprüft werden kann, ob die eigene Stimme gezählt und korrekt entschlüsselt wurde oder nicht [BBH$^+$17, BBB$^+$13].

ThreeBallot

Bei diesem elektronischen Wahlsystem besteht ein Stimmzettel aus drei Spalten, wobei jede einen eigenen Stimmzettel darstellt (siehe Abbildung 2.4). Der obere Teil des Stimmzettels besteht aus den Auswahlmöglichkeiten der KandidatInnen und der untere Teil beinhaltet eine ID. Die IDs sind zufällig gewählte Zahlen und dürfen auf keinem Stimmzettel und auch nicht in deren Spalten gleich sein. Die drei Spalten der Stimmzettel stellen später jeweils einen eigenen Stimmzettel dar. Die WählerInnen müssen beim Ausfüllen des Stimmzettels wie folgt vorgehen [Riv06]:

- Alle drei Spalten des Stimmzettels müssen berücksichtigt werden und für die Wahl sollte der Stimmzettel Zeile für Zeile abgearbeitet werden.
- Um für einen Kandidaten oder eine Kandidatin zu stimmen, müssen genau zwei der drei Felder in der Reihe des gewünschten Kandidaten oder der gewünschten Kandidatin ausgefüllt werden. Dabei ist die Auswahl, welche zwei das sind, nicht wichtig.
- Um gegen KandidatInnen zu wählen (also nicht dafür) muss genau einer der drei Kreise in der jeweiligen Reihe ausgefüllt werden (es ist nicht relevant, welcher der drei Kreise

ausgefüllt wird).

- Damit der Stimmzettel gültig ist, muss in jeder Reihe mindestens ein Kreis ausgefüllt werden. Andernfalls wird der Stimmzettel nicht akzeptiert.
- Es dürfen in keiner Reihe alle drei Kreise ausgefüllt werden. In diesem Fall wird der Stimmzettel nicht akzeptiert.
- Es darf insgesamt nur maximal für einen Kandidaten oder eine Kandidatin pro Amt gestimmt werden. Es ist aber auch erlaubt gegen alle KandidatInnen zu stimmen.
- Die Anzahl der pro Spalte ausgefüllten Kreise ist irrelevant.

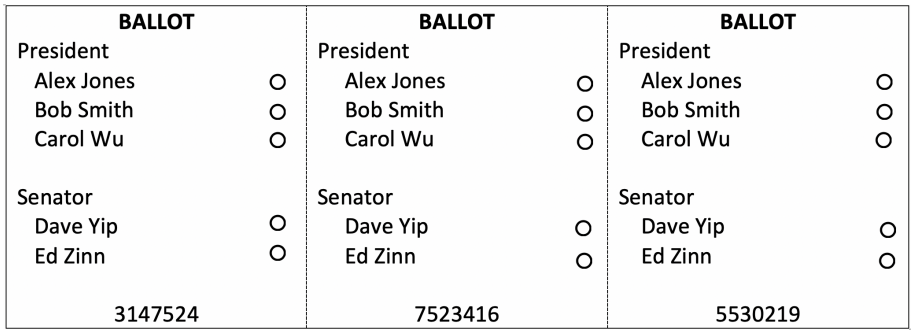

Abbildung 2.4: Ein ThreeBallot Stimmzettel nach [Riv06]

Wenn der Wähler oder die Wählerin den Stimmzettel ausgefüllt hat, wird über eine Maschine überprüft, ob dieser gültig ist. Sollte der Stimmzettel nicht korrekt ausgefüllt worden sein, so signalisiert das die Maschine und gibt an, wo der Fehler vorliegt. Ist der Stimmzettel gültig, wird ein Streifen unter die IDs gedruckt und der dreiteilige Stimmzettel wird zertrennt und ab dann als drei separate Stimmzettel betrachtet. Die Maschine, welche die Stimmzettel überprüft, speichert keine Daten. Als Quittung erhalten die WählerInnen eine Kopie von einem der drei Stimmzettel (zufällig gewählt). Diese Quittung und auch jeder einzelne Teil des Stimmzettels gibt auf Grund des Aufbaus dieses Wahlsystems keine Informationen über die Auswahl des Wählers bzw. der Wählerin preis. Die Stimmzettel kommen dann separat in die Wahlurne, wo sie noch einmal vermischt werden [Riv06].
Nach Ende der Wahl werden die Stimmen und eine Liste aller WählerInnen, welche an der Wahl teilgenommen haben, veröffentlicht. Mit Hilfe der Quittung kann überprüft werden, ob eine Stimme gezählt wurde oder nicht. Sollte die Stimme nicht gezählt worden sein, kann dies mit der Quittung in einem Wahlbüro gemeldet werden. In diesem Fall werden die Stimmen erneut gescannt und gezählt [Riv06].

Auszählung: Die Auszählung selbst findet wie gewohnt statt. Die Ergebnisse der KandidatInnen lassen sich auch durch gewöhnliches Auszählen bestimmen, nur dass die Summe der Stimmen mit der Anzahl der WählerInnen wächst. Ein Beispiel hierfür wäre:
Sei n die Anzahl der WählerInnen und A, B und C die KandidatInnen. Die KandidatInnen hätten bei einer normalen Wahl a, b bzw. c Stimmen erhalten. Wenn von dem ThreeBallot System ausgegangen wird, so haben die KandidatInnen $n+a, n+b$ bzw. $n+c$ „Stimmen“ (die Gesamtanzahl der für einen Kandidaten oder eine Kandidatin ausgefüllten Kreise) bekommen. Um die tatsächliche Anzahl der Stimmen zu erhalten, muss also einfach n vom Ergebnis subtrahiert werden.
Es ist zusätzlich auch möglich anstatt des Ausfüllens der Kreise ein Ranking durch das Angeben von Zahlen durchzuführen, damit eine Reihung entsteht (mehr dazu siehe [RS07]).

VAV-System

Ein System, welches dem ThreeBallot System sehr ähnlich ist, ist das VAV-System (Vote, Anti-Vote, Vote System). Ein Unterschied liegt darin, dass jeder der drei Teile des Stimmzettels schon als Vote oder Anti-Vote vorgegeben ist (siehe Abbildung 2.5). In der Gesamtauswahl muss ein Vote/Anti-Vote Paar existieren und zusätzlich noch die tatsächliche Auswahl getroffen werden. Das heißt es werden nur zwei Zeilen ausgefüllt. Die Votes und die Anti-Vote müssen so aufgeteilt werden, dass jede Spalte des Stimmzettels nur eine Auswahl beinhaltet. Nach dem Beispiel aus Abbildung 2.5 wären die ersten zwei Spalten des Stimmzettels also das Vote/Anti-Vote Paar und die tatsächliche Auswahl befindet sich in der letzten Spalte [RS07].

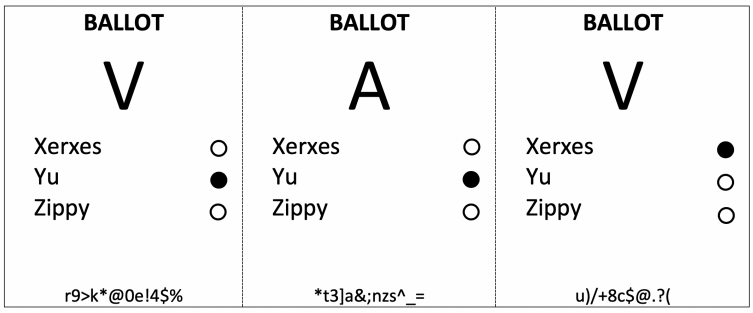

Abbildung 2.5: Ein VAV Stimmzettel nach [RS07]

Twin

Dieses System ist ein simples System, bei dem jeder Wähler / jede Wählerin einfach den Stimmzettel ausfüllt und diesen in einen Sammelkorb legt. Der Wähler / die Wählerin erhält eine Kopie eines zufälligen, bereits im Korb liegenden Stimmzettels als Quittung für die Wahl [RS07].
Zusätzlich existiert unter einem Feld zum Aufrubbeln noch eine ID. Diese bleibt unberührt bis der Stimmzettel von einer Maschine überprüft wird. Bei Gültigkeit wird der Stimmzettel wieder in den Korb gelegt und das Rubbelfeld wird dabei automatisch aufgekratzt. Dieses Prozedere wird so durchgeführt, dass die BenutzerInnen zwar sehen was passiert, die ID unter dem Rubbelfeld bleibt ihnen dabei aber trotzdem verborgen. Ein Nachteil bei diesem Wahlsystem liegt darin, dass die WählerInnen nicht verifizieren können, ob ihre Stimmen gezählt wurden [RS07].

2.2.2 Remote Systeme

Remotegrity

Remotegrity ist kein alleinstehendes elektronisches Wahlsystem, es kann mit einem Wahllokal-System (wie zum Beispiel Scantegrity) kombiniert werden. Folgende kryptographische Eigenschaften werden für das Verwenden von Remotegrity benötigt [ZCC+13]:

- Ein *verteiltes Schlüsselerzeugungs-Protokoll* (DKG) welches zur Erstellung von Schwellwert-Schlüsseln eines geheimen Seeds s verwendet wird. Diese Schlüssel sollen auf die Vertrauten der Wahl verteilt werden.

- Ein *Pseudo-Zufallszahlen Generator* (PRG(s)), um den Seed im Bezug auf die Pseudo-Zufälligkeit zu erweitern
- Eine *Commitment-Funktion* Comm(m,r), welche den Bezug zwischen der Nachricht m und dem Rauschfaktor r verbergen soll

Abbildung 2.6: Der Remotegrity Stimmzettel und die Autorisierungskarte nach [ZCC+13]

Für den Wähler bzw. die Wählerin selbst läuft eine Wahl folgendermaßen ab:
Den WählerInnen wird der Stimmzettel mit einer Autorisierungskarte (*Authorization Card*), welche eine Seriennummer, Authentifizierungscodes (zum Aufrubbeln), einen Bestätigungscode und einen Lock-in Code (zum Aufrubbeln) enthält, zugesendet. Vom Benutzer oder der Benutzerin muss beim Antreten der Wahl die Seriennummer der Autorisierungskarte auf der grafischen Benutzeroberfläche der Wahlseite eingegeben werden. Es wird vom System überprüft, ob diese Seriennummer schon einmal auf einem Bulletin Board (BB) veröffentlicht wurde [BBH+17, ZCC+13].
Sollte mit einem System wie Scantegrity gearbeitet werden, so muss der Benutzer oder die Benutzerin im nächsten Schritt den Auswahlcode, welcher neben dem gewählten Kandidaten oder der gewählten Kandidaten sichtbar wird, zusammen mit einem zufälligen Authentifizierungscode, auf der Website eingeben. Der Wähler oder die Wählerin wählt den Authentifizierungscode indem er oder sie zufällig eine der vier Flächen (siehe Abbildung 2.6) frei rubbelt, der Code erscheint dann darunter. Der Auswahlcode und der Authentifizierungscode werden dann gemeinsam auf dem Bulletin Board veröffentlicht [BBH+17, ZCC+13].
Die WahlhelferInnen überprüfen auch hier, ob dieser Authentifizierungscode schon einmal vorgekommen ist. Sollte die Stimme gültig sein, hängen die WahlhelferInnen zusätzlich den Bestätigungscode an und signieren die gesamte Stimme, welche die Seriennummer, den Auswahlcode, den Authentifizierungscode und den Bestätigungscode enthält. Sollte die Stimme ungültig sein, so wird sie als ungültig markiert und trotzdem signiert [ZCC+13].
Im nächsten Schritt überprüft der Wähler oder die Wählerin, ob die Stimme, welche auf dem Bulletin Board erscheint, korrekt ist und verifiziert den Bestätigungscode und die Signatur. Bei Gültigkeit gibt der Wähler oder die Wählerin den Lock-in Code ein.
Nachdem die Wahl beendet ist überprüfen die WahlhelferInnen, welche Lock-in Codes gültig sind und geben die gültigen Stimmen zur Auszählung weiter. Das Ergebnis wird anschließend veröffentlicht [ZCC+13].

Helios

Bei diesem System muss eine Internetseite besucht werden, damit der Wahlprozess durchgeführt werden kann. Zu allererst muss der Wähler/die Wählerin den Wahlprozess starten indem die Wahl, an welcher er oder sie teilnehmen möchte, ausgewählt wird. Dies findet über das sogenannte *Ballot Preparation System* (BPS) oder auch Stimmzettel-Vorbereitungssystem statt. Folgende Schritte werden bei dem Auswahlprozess durchgeführt [Adi08, BBH+17]:

- Wenn der Wähler oder die Wählerin die Auswahl getroffen und bestätigt hat, verschlüsselt das BPS die Auswahl und legt sich auf diese Verschlüsselung durch Vorzeigen eines Hashes über den Ciphertext fest.
- Der Wähler oder die Wählerin kann sich dafür entscheiden diesen Stimmzettel zu auditieren. Sollte dies der Wunsch des/der Wählenden sein, zeigt das BPS den Ciphertext und den Rauschfaktor, welcher für die Verschlüsselung verwendet wurde, an. Somit kann der/die WählerIn verifizieren, dass das BPS die Stimme korrekt verschlüsselt hat. Anschließend muss die Verschlüsselung erneut durchgeführt werden.
- Im Anschluss kann der/die WählerIn die Stimme *versiegeln*. Dafür verwirft das BPS den Rauschfaktor und die Klartext-Information, es bleibt nur der Ciphertext.
- Anschließend muss sich der/die Wählende authentisieren. Hierfür müssen die zur Verfügung gestellten persönlichen Zugangsdaten eingegeben werden. Bei erfolgreicher Authentisierung wird die verschlüsselte Stimme dem Wähler bzw. der Wählerin zugeschrieben.
- Das Helios Bulletin Board veröffentlicht den Namen des Wählers bzw. der Wählerin zusammen mit der verschlüsselten Stimme, was jeder sehen kann.
- Nachdem die Wahl abgeschlossen ist, werden die Stimmen mit Hilfe eines *Mixnets* gemischt und re-randomisiert und sowohl für das Mischen als auch für das Re-randomisieren ein Korrektheitsbeweis erstellt.
- Dieser Beweis wird überprüft und anschließend werden alle verschlüsselten Stimmen entschlüsselt. Es wird zu jeder Entschlüsselung ein Entschlüsselungs-Beweis erstellt und die Auszählung wird durchgeführt. Eine Bestätigung, dass die Stimme des Wählers oder der Wählerin eingegangen ist, wird per E-Mail versandt.
- Ein Auditor oder eine Auditorin kann die ganzen Daten herunterladen, um die Korrektheit der Wahl zu überprüfen.

Für die Durchführung des Mixnets wird das *Sako-Kilian Protokoll* [SK95] verwendet, welches auf der ElGamal Verschlüsselung basiert. Für den Beweis über die korrekte Entschlüsselung wird das *Chaum-Pedersen Protokoll* [CP92] verwendet [Adi08].

Selene

Bei diesem elektronischen Wahlsystem erhalten die WählerInnen eine Einladung zu Wahl. Sie bekommen bei der Registrierung bei der *Wahlbehörde* (W) eine *Tracker Nummer* für die Nachvollziehbarkeit, welche später dem Wähler oder der Wählerin helfen soll zu identifizieren, ob die eigene Stimme gezählt wurde, oder nicht. Der Wähler oder die Wählerin gibt seine oder ihre Auswahl ein und das Device des Wählers oder der Wählerin verschlüsselt die Stimme mit dem zur Wahl zugehörigen öffentlichen Schlüssel. Weiters wird vom Device eine Signatur über die verschlüsselte Stimme erstellt. Anschließend wird die verschlüsselte Stimme samt eines Proof of Knowledge des Klartextes an den Wahlserver gesendet. Die WählerInnen werden informiert, nachdem die Stimmen am Web Bulletin Board veröffentlicht wurden. Sie können dann die Korrektheit der Stimmen überprüfen [BBH+17, RRI16, BDS17].

Auch bei diesem elektronischen Wahlsystem wird für das Mixnet das *Chaum-Pedersen Protokoll* [CP92] verwendet.

sElect

sElect wird als Plattform-unabhängige Webapplikation implementiert. Die Idee dahinter ist, dass WählerInnen sich einen von mehreren Wahlservern aussuchen können über den sie wählen können. Das elektronische Wahlprotokoll arbeitet mit asymmetrischer Verschlüsselung und digitalen Signaturen [KSM+16].
Im folgenden sind die einzelnen Komponenten der Wahl aufgelistet [KSM+16]:

- Das *Bulletin Board*, wo die Ergebnisse veröffentlicht werden
- Die *WählerInnen*
- Die *Voter Supporting Devices* (VSD) oder auch WählerInnen-Devices, was zum Beispiel der Browser des Wählers oder der Wählerin ist
- Die *Authentifizierungs-Server* (AS), über die die WählerInnen authentifiziert werden
- Die *Mixing-Server*, welche für das Mischen der Stimmen benötigt werden
- Die *Wahlbehörde*

Bei sElect besteht eine Wahl aus mehreren Phasen:

Aufsetzungsphase In dieser Phase werden wichtige Parameter für die Wahl (wie zum Beispiel die Wahlidentifikation, KandidatInnenliste, Liste aller Wahlberechtigten,...) festgelegt und von der Wahlbehörde am Bulletin Board veröffentlicht. Außerdem führt jeder beteiligte Server (Mix Server und AS) den Schlüsselgenerierungs-Algorithmus durch, um die öffentlichen und privaten Schlüssel zu generieren. Die öffentlichen Schlüssel werden dann ebenfalls am Bulletin Board veröffentlicht [KSM+16].

Wahlphase Sollte eine wahlberechtigte Person eine Stimme abgeben wollen, so tut sie dies über das VSD. Es wird nach der Auswahl ein Bestätigungscode generiert, welchen sich der Wähler bzw. die Wählerin merken muss, da am Ende der Wahl die Auswahl/Bestätigungscode Paare veröffentlicht werden, sodass die WählerInnen überprüfen können, ob ihre Stimmen tatsächlich gezählt wurden. Wenn der Wähler oder die Wählerin seine/ihre Stimme abgegeben hat, werden die Auswahl und die Bestätigungscodes mit dem öffentlichen Schlüssel des letzten Mixing-Servers verschlüsselt. Das daraus resultierende Paar wird dann mit dem öffentlichen Schlüssel des vorletzten Mixing-Servers verschlüsselt. Dieses Prozedere wird wiederholt bis alle Schlüssel der Mixing-Server verwendet wurden und somit insgesamt m Verschlüsselungen (also eine Schichten-artige Verschlüsselung) durchgeführt wurden, wobei m die Anzahl der Mixing-Server ist. Das wird dann einem Authentifizierungs-Server vorgelegt, welcher die Stimme signiert, falls sie korrektes Format hat. Sollte ein Wähler oder eine Wählerin versuchen erneut zu wählen, wird dies vom Authentifizierungs-Server nicht akzeptiert. Nach der Wahlphase werden vom AS zwei Listen am Bulletin Board veröffentlicht. Eine davon enthält alle gültigen Stimmen (verschlüsselt) C_0 und die andere enthält alle WählerInnen, welche gültige Stimmen abgegeben haben [KSM+16].

Mixing Phase Als Input für den ersten Mixing-Server wird C_0 genommen. Der Server entschlüsselt C_0 mit seinem privaten Schlüssel, falls C_0 das richtige Format hat. Den Output signiert er und veröffentlicht das anschließend am Bulletin Board und gibt das Ergebnis dem nächsten Mixing-Server, der den gleichen Vorgang wiederholt. Dieses Prozedere wird fortgeführt, bis alle Mixing-Server an der Reihe waren [KSM+16].

Verifikationsphase Nachdem der letzte Schritt durchgeführt wurde, und somit C_m am Bulletin Board veröffentlicht wurde, werden zwei Verifikationsschritte durchgeführt. Bei der *WählerInnen basierten Verifikation* überprüfen die WählerInnen über ihre Verifikations-Codes ihre Auswahl am Bulletin Board. Sollte ein Wähler oder eine Wählerin seine/ihre Stimme nicht finden können, oder ein Fehler vorliegen, kann dieser/diese eine Beschwerde abgeben. Im zweiten Schritt wird die *VSD basierte Verifikation* durchgeführt. Dabei wird vom VSD automatisch überprüft, während der Wähler oder die Wählerin auf dem Device die Ergebnisse der Wahl abruft, ob die ursprünglich angegebene Stimme in C_m vorkommt. Sollte dies nicht der Fall sein, so wird versucht herauszufinden, welche Entität sich hier falsch verhalten hat. Es wird außerdem ein kryptographischer Beweis für falsches Verhalten des Servers erstellt [KSM+16].

Civitas

In diesem elektronischen Wahlsystem gibt es verschiedene Entitäten, welche unterschiedliche Rollen haben (siehe Abbildung 2.7) [SMH17, CCM08]:

- *Supervisor:*
  - administriert eine Wahl
  - setzt die Wahl auf
  - veröffentlicht den öffentlichen Schlüssel der Wahl
- *RegistratorIn (Registrar):*
  - gibt bekannt, wer zur Wahl zugelassen ist
  - sendet an die zugelassenen Personen einen öffentlichen Schlüssel und eine Kennungsnummer
- *Registrierungs-HelferInnen (Registration Tellers):*
  - generieren die Zugangsdaten, welche die WählerInnen für die Wahl verwenden
  - lassen den WählerInnen ihre Zugangsdaten zukommen
  - autorisieren die WählerInnen
- *Auszählungs-HelferInnen (Tabulation Tellers):*
  - entschlüsseln nach Ende der Wahl die Stimmen der WählerInnen
  - zählen die Stimmen aus

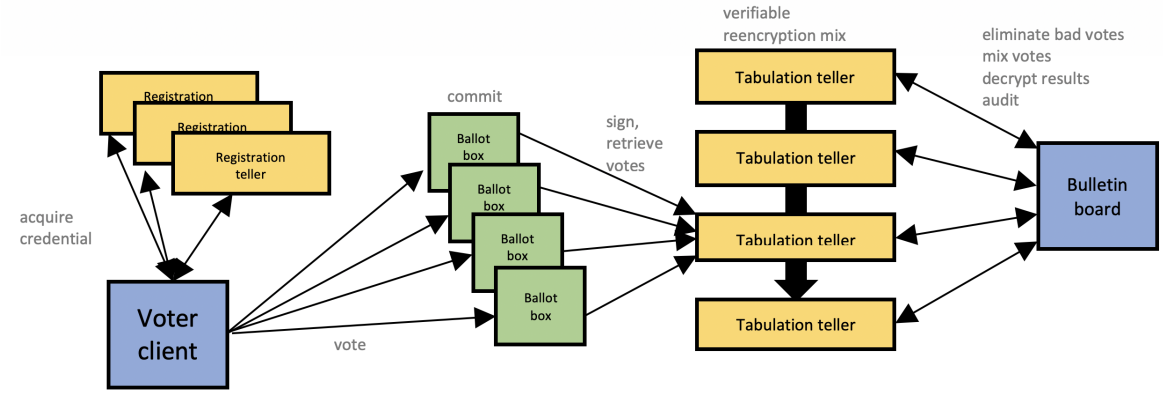

Abbildung 2.7: Der Civitas Wahlablauf nach [CCM08]

Die Wahl selbst kann in mehrere Phasen unterteilt werden [CCM08]:

Aufsetzungsphase Hier setzt der Supervisor die Wahl auf, indem die öffentlichen Schüssel der WahlhelferInnen auf ein Bulletin Board gepostet werden, um diese zu identifizieren. Außerdem veröffentlicht der Registrator oder die Registratorin das WählerInnenverzeichnis, wo die Kennungsnummern aller autorisierten WählerInnen und deren öffentliche Schlüssel enthalten sind. Jeder Wähler bzw. jede Wählerin hat in diesem System 2 Schlüsselpaare: einen Registrierungsschlüssel und einen Bestimmungsschlüssel. Die Auszählungs-HelferInnen generieren einen öffentlichen Schlüssel für ein verteiltes Verschlüsselungsverfahren und veröffentlichen diesen am Bulletin Board. Das heißt, wenn eine Nachricht mit diesem Schlüssel verschlüsselt wurde, werden alle Auszählungs-HelferInnen gebraucht, um diese Nachricht wieder zu entschlüsseln. Die Registrierungs-HelferInnen generieren Zugangsdaten, um die Stimmen anonym zu authentifizieren. Jeder Wähler/ jede Wählerin bekommt seine/ihre persönlichen Zugangsdaten [CCM08].

Wahlphase Die WählerInnen registrieren sich und erhalten ihre privaten Zugangsdaten. Jeder/jede der Registrierungs-HelferInnen authentifiziert einen Wähler oder eine Wählerin indem er/sie den Registrierungsschlüssel des Wählers/der Wählerin verwendet. Danach durchlaufen sie ein Protokoll unter Verwendung des Bestimmungsschlüssels des Wählers/der Wählerin. Um zu wählen verwendet der Wähler/die Wählerin seine/ihre privaten Zugangsdaten und gibt seine/ihre Stimme bei den Wahlurnen (Ballot Boxes) ab [CCM08].

Auszählungsphase Hierbei erhalten die Auszählungs-HelferInnen die Stimmen aus den Wahlurnen und die öffentlichen Zugangsdaten des Bulletin Board. Die Auszählungs-HelferInnen kontrollieren, ob alle Stimmen gültig sind und beseitigen jene, welche ungültig sind. Auch duplizierte Stimmen werden entfernt. Die Liste der abgegebenen Stimmen und die Liste der autorisierten Zugangsdaten werden durch eine Zufallspermutation mit Hilfe eines Mixnets anonymisiert. Auch unautorisierte Stimmen werden entfernt. Als letztes werden die verbliebenen Stimmen entschlüsselt und veröffentlicht, sodass jede Person die Auszählung selbst vornehmen könnte [CCM08].

Verifikation Zum Schluss muss jeder der Auszählungs-HelferInnen Beweise darüber veröffentlichen, dass sie dem Protokoll ehrlich gefolgt sind. Jeder kann diese Beweise überprüfen, sowohl während als auch nach der Wahl. Auch ein Wähler/eine Wählerin kann überprüfen, ob seine/ihre Stimme gezählt wurde, oder nicht [CCM08].

2.3 Vergleich der vorgestellten Systeme

In diesem Abschnitt wird ein kurzer Überblick darüber gegeben, inwiefern sich die vorgestellten Systeme aus IT-Sicherheits-Sicht unterscheiden. Hierfür werden einige der Faktoren aus 2.1 für die Beurteilung der Sicherheit von elektronischen Wahlsystemen betrachtet und die Systeme anhand dieser verglichen. Folgende Faktoren werden betrachtet:

- Integrität
- Privatsphäre
- Überprüfbarkeit
- Zwang zur Wahl
- Keine Belegbarkeit
- Wähler Nachweisbarkeit
- Korrektheit
- Benutzerfreundlichkeit (Usability)

Zusätzlich zu den aus Abschnitt 2.1 gewählten Eigenschaften wurde auch die *Benutzerfreundlichkeit* gewählt. Diese Kategorie soll eine grobe Einschätzung über die einfache Benutzbarkeit des Systems geben.
Aus Tabelle 2.1 kann entnommen werden, dass keines der betrachteten Systeme alle Bedingungen perfekt erfüllt. Es kann außerdem gezeigt werden, dass bei Remote Systemen das Erfüllen der *Zwang zur Wahl*-Bedingung nicht so einfach ist, wie bei Wahllokal-Systemen. Generell ist es schwer bei Remote Systemen eine hohe Sicherheitsstufe zu erreichen, ohne die Komplexität des Systems für den User oder die Userin zu erhöhen.

Tabelle 2.1: Vergleich der elektronischen Wahlsysteme aus dem IT-Sicherheitsaspekt

	Integrität	Privatsphäre	Überprüfbarkeit	Zwang	Keine Belegbarkeit	Nachweisbarkeit	Korrektheit	Usability
Prêt à Voter [RBHS10]	wenn alle Beteiligten ehrlich sind	Ja	Ja	Nein	Ja	Ja	Ja	Einfach
Scantegrity [CEC+08, BBH+17, CEC+08]	Ja	Nur durch Erweiterung	Ja	Ja	Ja	Ja	Ja	Einfach
STAR-Vote [AIK+16, BBB+13] [BBH+17]	Nein	Nur durch Erweiterung	Ja	Ja	Ja	Ja	Ja	Einfach
ThreeBallot [Riv06, BBH+17]	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Kompliziert
VAV [RS07]	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Kompliziert
Remotegrity [ZCC+13, BBH+17]	Ja	Nein	Ja	Nein	Abhängig von der Ausführung	Nur durch Erweiterung	Nein	Kompliziert
Helios [BBH+17, Adi08, BKV17]	Ja	Nur durch Erweiterung	Ja	Nein	Nein	Ja	Ja	Mittel
Selene [BBH+17, RRI16, JKK18, BDS17]	Nein	Nur durch Erweiterung	Ja	Ja	Ja	Ja	Nein	Mittel
sElect [KSM+16]	Nein	Ja	Ja	Nein	Nein	Ja	Ja	Einfach
Civitas [CCM08, SMH17, BBH+17]	Ja	Nur durch Erweiterung	Ja	Ja	Ja	Ja	Ja	Kompliziert

2.4 Zwei Beispiele für Anwendungen von elektronischen Wahlsystemen

In diesem Abschnitt werden zwei Beispiele für elektronische Wahlsysteme aus der Praxis angeführt. Hierfür wird als Beispiel Estland herangezogen, da dieses eines der ersten Länder war, welches ein Internet Wahlsystem eingeführt hat. Einige Länder haben dieses System sogar von ihnen übernommen [Aye17].

Im Gegensatz dazu wird ein Beispiel angeführt, welches zeigt, dass IT-Sicherheit in solchen Systemen eine wichtige Rolle spielt.

2.4.1 Estland

In Estland werden schon seit 2005 nationale Wahlen über elektronische Wahlsysteme durchgeführt. Die elektronischen Wahlen werden mit Hilfe von asymmetrischer Kryptographie durchgeführt. Es wird ein nationales PKI System genutzt, um die WählerInnen zu authentifizieren. Die WählerInnen verschlüsseln ihre Stimme und erstellen anschließend eine digitale Signatur über diese. Beides wird mit Hilfe einer Client-seitigen Software durchgeführt. Über eine App können die WählerInnen verifizieren, dass ihre Stimmen tatsächlich gezählt wurden [BBH+17, EE19].

Dieses System bietet keinen Beweis für eine korrekte Auszählung und keine Möglichkeit zu zeigen, dass die Stimmen korrekt aufgenommen wurden, wenn der Client unehrlich ist. Außerdem zeigte sich 2013, dass mit diesem System die Wahl manipuliert werden kann und dass es Schwachstellen bezüglich Schadsoftware auf der Client-Seite gibt. Außerdem ist dieses System auch für DDOS (Distributed Denial of Service) Attacken anfällig. Trotzdem wird ständig an der Verbesserung des Systems gearbeitet um dieses weiter verwendbar zu machen [BBH+17, EE19].

2.4.2 DC Digital Vote-by-Mail Service

Dies ist ein Wahlsystem, welches 2010 in Washington D.C. gestartet wurde. Die Idee dahinter war es den BewohnerInnen, welche zur Zeit der Wahl nicht in der Stadt waren, die Möglichkeit zu geben über das Internet zu wählen [SMH17].

DVBM (Digital Vote by Mail Service) ist eine Open-Source Webapplikation, welche in Ruby und Rails implementiert wurde und auf einem Apache Server mit einer MySQL Datenbank im Hintergrund läuft. Die Datenbank speicherte die verschlüsselten Stimmzettel und alle Informationen über die UserInnen [SMH17].

Die Authentifizierung fand über die WählerInnen ID und einen 16-stelligen Hexadezimal-PIN statt. Diesen PIN haben die WählerInnen zuvor per Post erhalten. Nach der Authentifizierung konnte ein PDF heruntergeladen werden, durch das die Stimme abgegeben werden konnte. Anschließend musste dieses File wieder auf den Server geladen werden, welcher es dann mit dem öffentlichen Schlüssel der Wahl verschlüsselt hat. Diese verschlüsselten Stimmzettel wurden bis zum Ende der Wahl in der Datenbank gespeichert. Am letzten Tag der Wahl bekam ein nicht mit dem Internet verbundener Computer alle Stimmzettel, um diese mit dem privaten Schlüssel der Wahl zu entschlüsseln und auszuzählen. [SMH17].

Eine große Schwachstelle in diesem System war, dass Zugriff zum ganzen System über den Upload eines infizierten PDF-Files erlangt werden konnte. Bevor das System online ging, erlaubten die Verantwortlichen in Washington, D.C. den BügerInnen das System auf Schwachstellen zu testen. Innerhalb von 48 Stunden konnten Wolchok, Wustrow, Isabel und Hal-

derman [WWI+12] Kontrolle über den gesamten Server erlangen. Sie konnten alle Stimmen ändern und auch alle geheimen Stimmzettel erhalten. Der Zugriff wurde über zwei Werktage hinweg nicht erkannt und wäre vermutlich auch länger verborgen geblieben, wenn sie nicht einen Hinweis hinterlassen hätten. Aus diesem Grund wurde dieses System nie wirklich für eine Wahl genutzt [SMH17, WWI+12].

2.5 Ausblick auf einen neuen Ansatz für ein elektronisches Wahlprotokoll

Aus dem Vergleich aus Abschnitt 2.3 kann gefolgert werden, dass Remote Systeme oft deutlich komplexer als Wahllokal-Systeme sind. Außerdem ist es bei gewissen Bedingungen schwieriger diese für ein Remote System als für ein System mit Wahllokal erfüllbar zu machen (zum Beispiel *Zwang zur Wahl*).
Durch das Betrachten unterschiedlicher Systeme zeigte sich außerdem, dass es viele verschiedene Herangehensweisen gibt, gewisse Sicherheitseigenschaften zu erfüllen. Jedes der Protokolle ist auf seine eigene Weise einzigartig und für unterschiedliche Zwecke besser geeignet als andere.
Im folgenden Teil dieser Masterarbeit wird ein weiterer Ansatz für ein elektronisches Wahlprotokoll vorgestellt. Bei diesem Protokoll liegt der Fokus auf den sogenannten *Attributen* (wie zum Beispiel die Stimme, das Geschlecht, der Wahlsprengel,...). Zusätzlich zu der Stimme können weitere, öffentliche (nicht geheime) Attribute in den Stimmzettel hineincodiert und später für das Erstellen von aussagekräftigen Wählerstatistiken verwendet werden.
Im ersten Schritt werden einige kryptographische Grundlagen wiederholt, um ein Signaturverfahren vorzustellen, welches den Fokus auf genau den Aspekt der Attribute gelegt hat. Basierend auf diesem Signaturverfahren kann anschließend ein Protokoll beschrieben werden, für das die in Abschnitt 2.1 angeführten Sicherheitsbedingungen erneut überprüft werden.

3 Kryptographische Grundlagen

Im folgenden Abschnitt wird eine restriktive verblindete Variante der AGHO Signatur, welche als Grundlage für diese Masterarbeit herangezogen wird, ausgearbeitet. Die unverblindete AGHO Signatur wurde erstmals von Abe, Groth, Haralambiev und Ohkubo [AGHO11] vorgestellt. Um genauer auf das Verfahren eingehen zu können, werden im ersten Schritt die benötigten kryptographischen Grundlagen wiederholt und beschrieben.

3.1 Wiederholung kryptographischer Grundlagen

3.1.1 RSA Blindsignatur

Für die Beschreibung des RSA Verschlüsselungsverfahrens und der RSA Signatur werden folgende Voraussetzungen benötigt [Ber01]:

- seien p und q zwei Primzahlen (512 Bit)
- sei $n = p \cdot q$ und $\varphi(n) = (p-1) \cdot (q-1)$ die Ordnung von n
- sei e eine zufällige Zahl mit $e \in \mathbb{N}$ und $1 < e < \varphi(n)$ und e ist teilerfremd zu $\varphi(n)$ also $ggT(e, (p-1) \cdot (q-1)) = 1$
- sei d mit $1 < d < (p-1) \cdot (q-1)$ so, dass $e \cdot d \bmod \varphi(n) = 1$

So ist (e, n) bzw. e der öffentliche Schlüssel und d der private Schlüssel. Die Parameter $p, q, \varphi(n)$ müssen geheim bleiben [Ber01].
In Abbildung 3.1 ist das RSA Signaturverfahren ersichtlich. Hierbei wird, umgekehrt als bei der Verschlüsselung, mit dem privaten Schlüssel des Unterzeichners/der Unterzeichnerin eine Signatur über eine Nachricht erstellt, welche dann von jeder Person, die den öffentlichen Schlüssel des Unterzeichners/der Unterzeichnerin hat, verifiziert werden kann [Ber01].

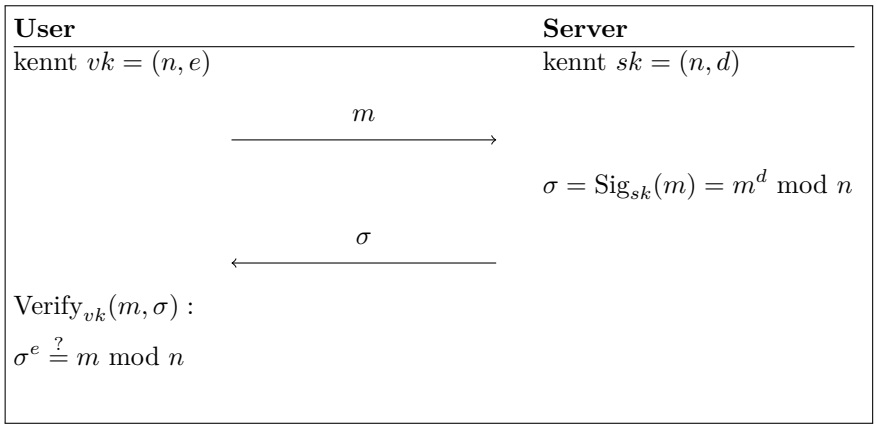

Abbildung 3.1: Ablauf RSA Signatur

Wenn verhindert werden soll, dass der Unterzeichner/die Unterzeichnerin Informationen aus der Nachricht, welche er/sie signieren soll, erhält (*Unlinkbarkeit*), so kann das Verfahren

der Blindsignatur gewählt werden. Hierfür wird nicht die tatsächliche Nachricht, sondern eine *verblindete* Version $\bar{m}$ der Nachricht, gesendet, um diese signieren zu lassen [Gre].
Das heißt es wird ein Rauschfaktor (Blindfaktor) $r \xleftarrow{\$} \mathbb{Z}_n$ gewählt und die Nachricht mit diesem folgendermaßen verblindet: $\bar{m} = r^e \cdot m$. Aus Abbildung 3.2 kann der Verlauf der verblindeten RSA Signatur entnommen werden [Gre].

User		**Server**
kennt $vk = (n, e)$		kennt $sk = (n, d)$
$r \xleftarrow{\$} \mathbb{Z}_n$		
$\bar{m} = r^e \cdot m \bmod n$		
	$\bar{m} \longrightarrow$	
		$\bar{\sigma} = \mathrm{Sig}_{sk}(\bar{m}) = \bar{m}^d \bmod n$
	$\longleftarrow \bar{\sigma}$	
Entblinden:		
$\sigma = \bar{\sigma}/r$		
$\mathrm{Verify}_{vk}(m, \sigma)$:		
$\sigma^e \stackrel{?}{=} m \bmod n$		

Abbildung 3.2: RSA Blindsignatur-Verfahren

Für die Verifikation muss somit nur noch durch den Rauschfaktor dividiert werden, um wieder die tatsächliche Signatur zu erhalten, da:

$$\begin{aligned} \bar{\sigma} = Sig_{sk}(\bar{m}) &= \bar{m}^d \\ &= (r^e \cdot m)^d = r^{e \cdot d} \cdot m^d = r \cdot m^d \\ \text{und somit: } \bar{\sigma}/r &= (r \cdot m^d)/r = m^d \end{aligned} \tag{3.1}$$

Die Verifikation kann dann wie bei der normalen RSA Signatur durchgeführt werden [Gre].

3.1.2 Klassisches ElGamal Verschlüsselungsverfahren

Das ElGamal Verschlüsselungsverfahren ist eine Verschlüsselung aus der Public Key Kryptographie (PKC), welches auf diskreten Logarithmen beruht. Für die ElGamal Verschlüsselung werden folgende Voraussetzungen benötigt [Ber02]:

- sei q Primzahl, g ein Generator mod q.
- sei m eine zu verschlüsselnde Nachricht mit $0 \leq m < q$.
- Sei sk mit $1 < sk < q - 1$ zufällig gewählt, der private Schlüssel
- sei $pk = g^{sk} \bmod q$ der dazugehörige öffentliche Schlüssel

Dann sieht die ElGamal Verschlüsselung folgendermaßen aus [Ber02]:

$$\begin{aligned} 1 \leq o \leq q-1 \text{ zufällig} \\ k = pk^o \bmod q \\ c_1 = g^o \bmod q \\ c_2 = k \cdot m \bmod q \\ \text{Enc}_{pk}(m) = (c_1, c_2) \end{aligned} \tag{3.2}$$

Um einen ElGamal Ciphertext wieder zu entschlüsseln sind folgende Schritte notwendig [Ber02]:

$$\begin{aligned} k = c_1^{sk} \bmod q \\ m = c_2 \cdot k^{-1} \bmod q \end{aligned} \tag{3.3}$$

3.1.3 Elliptische Kurven

Da zur Gewährleistung von ausreichender Sicherheit in einem kryptographischen Verfahren nicht einfach die Bitlängen immer weiter erhöht werden können, wurde nach einer alternativen Lösung gesucht. Die Elliptic Curve Kryptographie (ECC) (bzgl. ElGamal - Gruppe) basiert auf dem diskreten Logarithmus Problem, was bekanntlich ein schweres mathematisches Problem ist. Eine elliptische Kurve ist folgendermaßen definiert [HL]:

Seien $a, b \in \mathbb{Z}$. Dann ist eine elliptische Kurve die Menge aller Tupel (x, y) (oder auch Punkte), welche folgende Gleichung erfüllen [HL]:

$$E : y^2 = x^3 + ax + b \tag{3.4}$$

Auf dieser elliptischen Kurve müssen nun Rechenoperationen und Gruppenaxiome definiert werden, um mit dieser arbeiten zu können [HL]:

- Das *neutrale Element* wird als der Punkt im Unendlichen $\underline{\mathcal{O}}$ definiert
- Für das neutrale Element gilt außerdem: $\underline{P} + \underline{\mathcal{O}} = \underline{\mathcal{O}} + \underline{P} = \underline{P}$
- Die *Punktaddition*, also für $\underline{A} + \underline{B} = \underline{C}$ mit $\underline{A}, \underline{B}, \underline{C} \in E$ ist der dritte Schnittpunkt der Kurve mit einer Geraden, welche durch die beiden Punkte $\underline{A}$ und $\underline{B}$ verläuft. Das bedeutet, um diesen Schnittpunkt zu errechnen, müssen folgende Schritte durchgeführt werden:
  - Errechnen der Geradengleichung ($y = kx + d$) für die Gerade g, welche beim Verbinden von $\underline{A}$ und $\underline{B}$ entsteht
  - Nun ist der Punkt, $\underline{C}$ auf E gesucht, welcher auch ein Punkt auf g ist. Daher muss im nächsten Schritt g und E gleichgesetzt werden.
  - Im letzten Schritt muss, um die Koordinaten herauszufinden, ein Koeffizientenvergleich durchgeführt werden.

  Allgemein lässt sich der Punkt folgendermaßen berechnen:

$$\begin{aligned} x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2 \\ y_3 = \frac{y_2 - y_1}{x_2 - x_1}(x_1 - x_3) - y_1 \end{aligned} \tag{3.5}$$

- Die *Punktverdoppelung* $2 \cdot \underline{A} = \underline{C}$ für $\underline{A}, \underline{C} \in E$ wird folgendermaßen durchgeführt

  - Die Tangente am Punkt $\underline{A}$ wird berechnet
  - Der Schnittpunkt $\underline{C}'$ der Tangente mit der Kurve wird ermittelt
  - Dieser Punkt $\underline{C}'$ muss um die x-Achse gespiegelt werden, um $\underline{C}$ zu erhalten.

  Verallgemeinert bedeutet das:

$$\begin{aligned} x_3 &= \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1 \\ y_3 &= \left(\frac{3x_1^2 + a}{2y_1}\right)(x_1 - x_3) - y_1 \end{aligned} \tag{3.6}$$

- Die Skalarmultiplikation ergibt sich durch wiederholtes Anwenden der Gruppenaddition: $n \cdot \underline{P} = \underbrace{\underline{P} + \underline{P} + ... + \underline{P}}_{n\text{-mal}}$. Ein effizienteres Verfahren für die skalare Multiplikation ist das sogenannte *Double-And-Add-Verfahren*.

Wie in diesem Kapitel wird auch in den folgenden Kapiteln die Punktnotation $\underline{A} = \begin{pmatrix} A_x \\ A_y \end{pmatrix}$ verwendet werden, um Punkte auf einer elliptischen Kurve zu beschreiben.

3.1.4 ElGamal Verschlüsselung auf elliptischen Kurven

Äquivalent zur Bezeichnung aus den Abschnitten 3.1.2 und 3.1.3 seien folgende Voraussetzungen gegeben:

- Sei $E : y^2 = x^3 + ax + b \bmod q$ die Weierstraß-Form einer elliptischen Kurve über einem Primzahlkörper und q eine Primzahl
- Sei $\underline{g} = \begin{pmatrix} g_x \\ g_y \end{pmatrix}$ ein Kurvenpunkt primer Ordnung $r \bmod q$
- Sei $\underline{m}$ die zu verschlüsselnde Nachricht (welche zum Beispiel als Punkt auf E realisiert sein kann)
- Sei der private Schlüssel $sk \xleftarrow{\$} 1, ..., q-1$ zufällig
- und der öffentliche Schlüssel $\underline{pk} = \underline{g}^{sk}$

Dann ist die ElGamal Verschlüsselung über elliptische Kurven folgendermaßen definiert:

$$\begin{aligned} o &\xleftarrow{\$} \{1, ..., q-1\} \text{ zufällig} \\ \underline{k} &= \underline{pk}^o \bmod q \\ \underline{c_1} &= \underline{g}^o \bmod q \\ \underline{c_2} &= \underline{k} \cdot \underline{m} \bmod q \\ \mathrm{Enc}_{\underline{pk}}(\underline{m}) &= (\underline{c_1}, \underline{c_2}) \end{aligned} \tag{3.7}$$

wobei die Potenz der Skalarmultiplikation und die Multiplikation der Punktaddition gemäß elliptischer Kurven entspricht.
Für die Entschlüsselung ergibt sich folgendes:

$$\begin{aligned} \underline{k} &= \underline{c_1}^{sk} \bmod q \\ \underline{m} &= \underline{c_2} \cdot \underline{k}^{-1} \bmod q \\ \text{bzw. } \underline{m} &= \underline{c_2} \cdot \underline{c_1}^{-sk} \bmod q \end{aligned} \tag{3.8}$$

3.2 Zero-Knowledge Proof

3.2.1 Allgemeines

Bei einem Zero-Knowledge Proof geht es darum, dass eine *Beweiserin (Prover)* P einen *Verifizierer* V davon überzeugen möchte, dass sie ein gewisses Geheimnis kennt. Dieses Geheimnis kann zum Beispiel eine Lösung zu einem bestimmten Problem sein. P möchte aber nicht, dass V dadurch irgendwelche Informationen über dieses Geheimnis und insbesondere nicht das Geheimnis selbst erhält [Dre].
Da V nicht ohne weiteres glauben wird, dass P das Geheimnis kennt, muss ein Beweis darüber erstellt werden, um V davon zu überzeugen: der sogenannte *Zero-Knowledge Proof* [Dre].
Ein Zero-Knowledge Proof ist also ein Beweis darüber, dass P eine Lösung zu einem Problem kennt, ohne durch diesen Beweis die Lösung oder Informationen über diese zu verraten. Das kann zum Beispiel eine Rechnung sein, die nur dann gelingen kann, wenn P einen bestimmten Parameter kennt [Dre].

Im Allgemeinen werden folgende Eigenschaften von Zero-Knowledge Verfahren erfüllt [Dre]:

- *Vollständigkeit:* ehrliche Beweiserinnen werden von Verifizierern akzeptiert
- *Korrektheit:* unehrliche Beweiserinnen werden von Verifizierern nicht akzeptiert
- *Zero-Knowledge:* Der Verifizierer bekommt durch den Beweis keinen Wissenszuwachs

Um dieses Verfahren etwas zu veranschaulichen wird oft *Ali Babas Höhle* als Beispiel herangezogen. In dem Beispiel geht es darum, dass sich in einer Höhle zwei Gänge befinden. Diese lassen sich am Ende durch eine verschlossene Tür verbinden (siehe Abbildung 3.3). Die Beweiserin, in diesem Fall *Alice*, möchte einem Verifizierer, *Bob*, beweisen, dass sie das Passwort zu dieser Türe kennt, ohne Bob tatsächlich das Passwort zu verraten [Dre].
Dazu kann Alice folgendermaßen vorgehen: Alice geht in die Höhle und wählt zufällig einen der beiden Gänge aus. In diesen Gang geht sie bis zur Türe und teilt Bob mit, dass er die Höhle jetzt betreten darf. Bob bleibt vor der Abzweigung stehen, kann also nicht sehen in welchem Gang sich Alice befindet. Er ruft Alice nun zu, über welchen der beiden Wege sie die Höhle verlassen soll: links oder rechts. Die Auswahl trifft Bob dabei zufällig (dies kann zum Beispiel durch einen Münzwurf geschehen). Je nachdem, ob Alice sich bereits auf der richtigen Seite der Höhle befindet, tritt sie ohne die Türe zu durchqueren aus der Höhle oder durchquert mit Hilfe des Wissens über das Passwort die Türe, um auf die andere Seite zu gelangen. Bob wird nun sehen, dass Alice aus dem richtigen Gang der Höhle kommt [Dre].
Er wird allerdings folgende Bedenken äußern: Was, wenn Alice einfach durch Zufall den richtigen Gang gewählt hat und in Wirklichkeit das Passwort gar nicht kennt? Daher wird er darauf bestehen diesen Versuch so lange zu wiederholen, bis er davon überzeugt ist, dass Alice die Wahrheit sagt [Dre].
Dass Alice durch Zufall den richtigen Weg errät ist nicht schwer. In diesem Fall läge die Wahrscheinlichkeit bei 50%. Wenn aber mehrere Durchläufe getätigt werden, so sinkt die Wahrscheinlichkeit, dass Alice in jeder Runde den Weg, welchen Bob ihr vorsagt, richtig erraten kann auf $\frac{1}{2^n}$ mit n als die Anzahl der Runden. Somit liegt die Wahrscheinlichkeit eine unehrliche Beweiserin nicht zu entdecken bei $1 - \frac{1}{2^n}$ [Dre].
Bob wird also den Versuch so lange wiederholen bis die Wahrscheinlichkeit, dass Alice durch Zufall so oft hintereinander den richtigen Gang gewählt hat, gering genug ist [Dre].

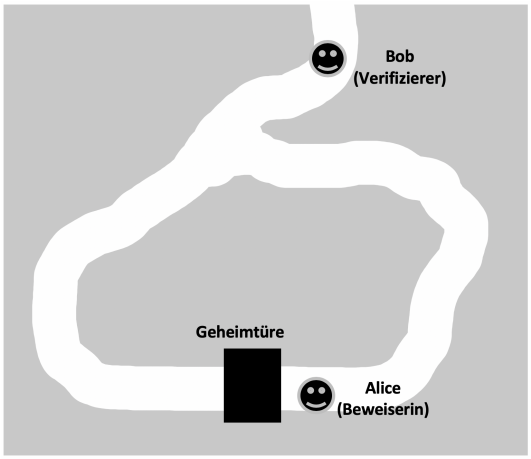

Abbildung 3.3: Ali Babas Höhle nach [Dre]

Simulierbarkeit

Um anhand des Höhlenbeispiels zu verstehen, was Simulierbarkeit bedeutet, kann von folgender Situation ausgegangen werden: Alice möchte oben beschriebenes Spiel noch einmal spielen, allerdings diesmal mit *Carl*. Anstatt dass Alice Carl vor die Höhle bittet, zeigt sie ihm ein Video davon, wie Alice jedes Mal aus dem richtigen Höhleneingang kommt. Der Unterschied besteht darin, dass sich Bob und Alice vor jedem Durchgang absprechen und entscheiden, in welchen Gang Alice gehen soll. Carl kann das auf dem Video nicht sehen. Er sieht nur, dass Alice jedes Mal aus dem richtigen Ausgang kommt [Dre].
Es gibt eine weitere Möglichkeit das Vorgehen zu simulieren, ohne, dass Bob und Alice sich absprechen. Wenn Alice sich im falschen Gang befindet, so wird dieser Teil des Videos einfach herausgeschnitten [Dre].
Wichtig ist, dass die simulierte Variante, egal welche der beiden genannten, nicht von einem echten Versuch unterschieden werden kann. Carl kann auf dem Video nicht erkennen, ob der Versuch tatsächlich durchgeführt wurde, oder ob Alice das Passwort gar nicht kennt. Diese Eigenschaft zeigt, dass es sich bei dem Beweis um einen Zero-Knowledge Proof handelt. Es kann ein gefälschtes Video ohne Kenntnis des Geheimnisses erstellt werden und das bedeutet, es kann auch aus der Durchführung eines echten Beweises keine Information über das Geheimnis erlangt werden [Dre].
Es kann also ein Video, welches zeigt, dass dieses Beispiel mit einem ehrlichen Beweiser durchgeführt wird nicht von einem Video unterschieden werden, indem ein unehrlicher Beweiser die Höhle betritt. Das bedeutet auch, dass jeder Algorithmus, welcher als Input echte Interaktionen bekommt auch mit gefälschten Interaktionen gefüttert werden kann, da diese ununterscheidbar sind [Dre].

3.2.2 Ein Beispiel für ein NP vollständiges Problem - Die Dreifärbbarkeit eines Graphen

Ein weiteres sehr bekanntes Beispiel, um die Zero-Knowledge Proofs zu erklären, aber auch ein beliebtes Beispiel in der Mathematik, ist jenes des dreifärbbaren Graphen. Solch ein Graph kann folgendermaßen definiert werden:
Sei $f : X \to \{0, 1, 2\}$ die Funktion, welche die Färbung eines Knotens des Graphen angibt. Dann heißt ein Graph *dreifärbbar*, wenn für alle benachbarten Knoten v_1, v_2 gilt $f(v_1) \neq$

$f(v_2)$, die Färbung der benachbarten Knoten also nicht gleich ist. Ein Beispiel für einen dreifärbbaren Graphen ist in Abbildung 3.4 gegeben und eine Lösung, also eine Dreifärbung dieses Graphen, ist in Abbildung 3.5 gegeben. Das Problem des dreifärbbaren Graphen ist ein NP-vollständiges Problem, für die Realität also durchaus ein interessantes Beispiel.

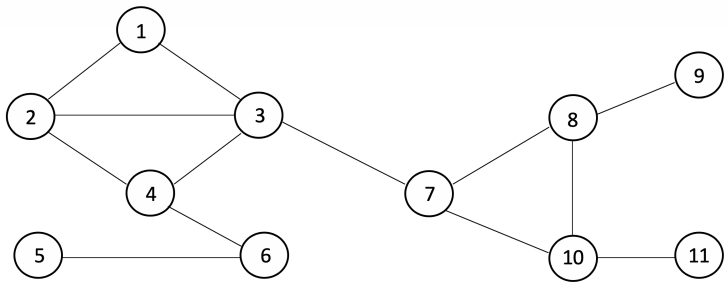

Abbildung 3.4: Ein dreifärbbarer Graph nach [Gre14]

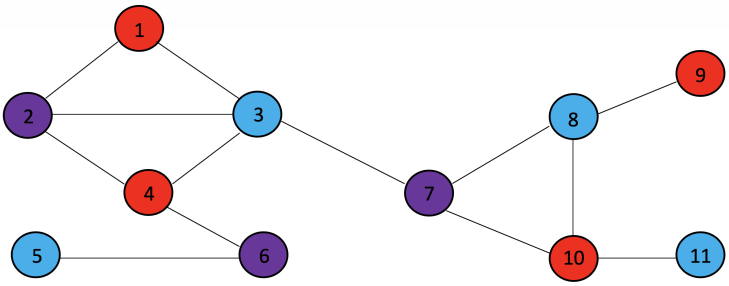

Abbildung 3.5: Eine mögliche Dreifärbung des Graphen aus Abbildung 3.4 nach [Gre14]

Sei also die Annahme getroffen, dass ein Beweiser P eine Lösung für die Dreifärbbarkeit gefunden hat. Da P die Lösung selbst nicht, und auch keine weiteren Informationen preisgeben möchte, aber V trotzdem beweisen möchte, dass er die Lösung kennt, wird hier ein Zero-Knowledge Proof gebraucht. Dieser funktioniert folgendermaßen [Gre14]:

- P zeichnet eine Repräsentation des dreigefärbten Graphen auf. Über jeden Knoten wird ein *Hütchen* gesetzt, um die vollständige Lösung zu verschleiern (wie in Abbildung 3.6).
- Unter diesen Hütchen könnte sich eine tatsächliche Dreifärbung des Graphen, oder auch eine zufällige Färbung befinden, wenn P unehrlich ist.
- Im nächsten Schritt wählt V eine Kante e des Graphen und darf sich die dazugehörigen Knoten (v_1, v_2) anschauen (das *Opening* - siehe Abbildung 3.7):
  - Sollten die Knoten gleich gefärbt sein, ist klar, dass P unehrlich ist.
  - Sollten die Knoten ungleich gefärbt sein, ist es möglich, dass P tatsächlich die Lösung kennt, es muss aber nicht sein. Wie auch in dem Höhlenbeispiel könnte P genauso auch geraten haben. Die Wahrscheinlichkeit, dass P mit einer Lüge davon kommt liegt bei $(e-1)/e$ mit e als Anzahl der Kanten.
- Da P im zweiten Fall trotzdem noch ein Betrüger sein könnte, wird das oben beschriebene Prozedere einfach wiederholt. Damit V nichts über die Lösung (also die Färbung) lernt, wird eine zufällige, geheime Permutation auf die ursprüngliche Färbung angewen-

det. So wäre zum Beispiel

$$\pi : \{rot, grün, blau\} \mapsto \{grün, blau, rot\} \tag{3.9}$$

eine gültige Permutation.

- Durch dieses Vorgehen sinkt die Wahrscheinlichkeit, dass P ein Betrüger ist bei Korrektheit auf $(e-1)/e \cdot (e-1)/e$.
- Das Prozedere kann so lange wiederholt werden, bis die Wahrscheinlichkeit, dass P ein Betrüger ist und nicht erwischt wird für V hinreichend klein ist. 100%ige Sicherheit wird V darüber nie haben.

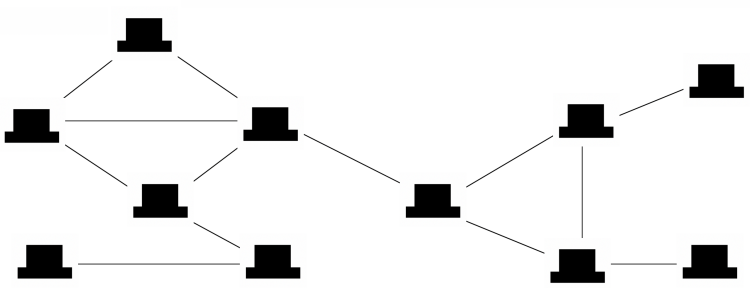

Abbildung 3.6: Verstecken der Lösung unter Hütchen nach [Gre14]

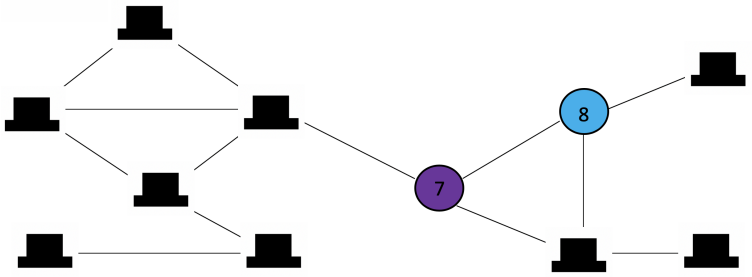

Abbildung 3.7: Das Opening von einer Kante nach [Gre14]

Wichtig zu erwähnen ist, dass die Chancen einen Betrüger zu entdecken am größten sind, wenn die Kanten von V zufällig ausgewählt werden, um Unvorhersagbarkeit der Antworten zu erzielen [Gre14]. Durch das Durchführen von mehreren Runden kann die Chance, dass P die Lösung errät stark reduziert werden, sodass die Wahrscheinlichkeit, dass P die Wahrheit sagen muss immer größer wird.

3.2.3 Schnorrscher Zero-Knowledge Proof

Eine spezielle Art des Zero-Knowledge Proofs ist jene, welche auf diskreten Logarithmen beruht (Schnorrscher Zero-Knowledge Proof of Knowledge - siehe auch Abbildung 3.8).
P kennt einen zu einem öffentlich bekannten Y stehenden Exponenten x und möchte V davon überzeugen, dass er x kennt, ohne zusätzliche Informationen preis zu geben. V kennt nur g und $Y = g^x$ wobei g ein Generator ist [Dre].
P überlegt sich eine zufällige Nonce ν (dies ist ein einmalig verwendeter Wert, um Wiederholbarkeit auszuschließen) und berechnet $N = g^\nu$. Durch die Berechnung von N legt sich P auf ν fest und kann diesen Wert im Nachhinein nicht mehr ändern. Dieser Wert, N, heißt

dann das Commitment (so wie das Zeichnen des Graphen und das Setzten der Hütchen im vorherigen Beispiel). P sendet dann N an V [Dre].
Im nächsten Schritt wählt V eine Challenge und sendet diese an P zurück, welcher dadurch die passende *Response* $r = ch \cdot x + \nu$ berechnet. Das ist nur möglich, wenn P das geheime x kennt. Die errechnete Response schickt P an V zur Verifikation zurück. Dieser Überprüft die Response folgendermaßen [Dre]:

$$\begin{aligned} r \stackrel{?}{=} ch \cdot x + \nu \Leftrightarrow g^r &\stackrel{?}{=} g^{ch \cdot x + \nu} \\ &= g^{ch \cdot x} \cdot g^{\nu} \\ &= g^{x \cdot ch} \cdot g^{\nu} \\ &= (g^x)^{ch} \cdot N \\ &= Y^{ch} \cdot N \end{aligned} \tag{3.10}$$

Die letzteren Werte sind V alle bekannt. Somit kann die Korrektheit der Aussage überprüft werden, ohne ν oder x zu kennen oder Informationen darüber zu erhalten.

BeweiserIn		**VerifiziererIn**
kennt (g, x)		kennt $Y = g^x$
$\nu \xleftarrow{\$} \mathbb{Z}_p$ $N = g^{\nu}$		
	$N \longrightarrow$	
		$ch \xleftarrow{\$} \mathbb{Z}_p$
	$\longleftarrow ch$	
$r = ch \cdot x + \nu$		
	$r \longrightarrow$	
		Verify: $g^r \stackrel{?}{=} Y^{ch} \cdot N$

Abbildung 3.8: Schnorrscher Zero-Knowledge Proof

Auch bei dieser Form des Zero-Knowledge Proofs sollten bei einem kleinen Challenge-Raum (wie zum Beispiel einem binären Raum $\{0, 1\}$) mehrere Runden durchgeführt werden, um die Wahrscheinlichkeit, dass P ein Betrüger ist, zu minimieren. Sollte der Challenge-Raum (zum Beispiel der gesamte Exponentenraum) bereits groß genug sein, ist die Wahrscheinlichkeit, die Challenge richtig zu erraten, bereits kryptographisch klein und es sind keine Wiederholungen notwendig. Sollte P das Geheimnis tatsächlich kennen, so wird es ihm in jeder Beweisrunde gelingen, die Challenge korrekt zu lösen. Ein Beweiser, welcher das Geheimnis aber nicht kennt, also unehrlich ist, und V von einer falschen Behauptung überzeugen will, kann in so einem Beweis eine korrekte Antwort liefern, wenn er die Challenge im Voraus richtig errät (durch Simulation) [Dre].

Die Eigenschaft der Simulierbarkeit kann für den Schnorrschen Zero-Knowledge Proof folgendermaßen gezeigt werden:

- P kennt den Generator g und das öffentliche Y aber nicht den geheimen Exponenten x
- P wählt/rät eine Challenge $ch \leftarrow \{\}$
- P wählt einen Rauschfaktor $r \leftarrow \{\}$
- P berechnet N, sodass $g^r = Y^{ch} \cdot N$
- P gibt die erhaltenen Werte in folgender Reihenfolge aus (N, ch, r).

Da der Beweiser bei der Ausgabe die Reihenfolge der erzeugten/errechneten Werte vertauscht, sieht es von außen so aus, als hätte er den Beweis korrekt durchgeführt. Das bedeutet also, dass die Ausgaben des Programms nicht von den Aussagen eines echten x-Besitzers unterschieden werden können, was die Zero-Knowledge Eigenschaft ausmacht.

3.2.4 Schnorrscher nicht-interaktiver Zero-Knowledge Proof

Um diese Form des Zero-Knowledge Proofs zu erläutern wird exemplarisch der Schnorrsche nicht-interaktive Zero-Knowledge Proof (NIZK) herangezogen.
Dabei wird vom gleichnamigen interaktiven Zero-Knowledge Proof ausgegangen und eine sogenannte *Fiat-Shamir* Transformation durchgeführt. Das bedeutet, es wird anstelle der zufälligen Wahl einer Challenge durch eine interaktive Verifiziererin, eine kryptographisch sichere Hashfunktion zur Erzeugung von ch verwendet. Mit obiger Notation kann die Challenge zum Beispiel die Form $ch = H(g||N||Y)$ haben, wobei H die Hashfunktion ist [Hao17, Dre]. Durch das Verwenden der Hashfunktion, welche die eigenständige Berechnung der Challenge erlaubt, wird V für die Durchführung des Zero-Knowledge Proofs nicht mehr benötigt und der Beweis wird somit *nicht-interaktiv*. Es reicht auch hier ein einziger Durchgang, da der Challengeraum (die Menge aller möglichen Hashes für die Hashfunktion H) so groß ist, dass die Wahrscheinlichkeit eine Challenge vorher zu erraten fast 0 ist.

Um das Verfahren anhand eines Beispiels zu zeigen, nehmen wir an, dass ein Beweiser P eine Verifiziererin V davon überzeugen will, dass er einen geheimen Exponenten x kennt, sodass gilt: $g^x = Y$, wobei g und Y bekannt sind. P wird bei einem nicht-interaktiven Zero-Knowledge Proof folgendermaßen vorgehen:

- Sei $\nu \xleftarrow{\$} \{\}$
- berechne $N = g^\nu$
- berechne die Challenge $ch = H(N)$
- berechne die Response $r = ch \cdot x + \nu$
- Ausgabe: $\pi = (ch, r)$
- Für die Verifikation kann V einfach folgendes überprüfen:

$$\begin{aligned} g^r &= g^{ch \cdot x + \nu} = Y^{ch} \cdot N \\ ch &\stackrel{?}{=} H(g^r \cdot Y^{-ch}) \end{aligned} \tag{3.11}$$

3.3 Die AGHO Signatur

Die AGHO (Abe, Groth, Haralambiev und Ohkubo [AGHO11]) Signatur ist ein Verfahren, welches auf bilinearen Gruppen (oder auch *Pairings*) arbeitet. Bilineare Gruppen haben die Form eines Tupels $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h)$ und erfüllen folgende Eigenschaften [AGHO11]:

- $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ sind Gruppen primer Ordnung p

- $e: \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ ist eine bilineare Abbildung, sodass $\forall U \in \mathbb{G}_1,\ \forall V \in \mathbb{G}_2,\ \forall a, b \in \mathbb{Z}_R : e(U^a, V^b) = e(U, V)^{ab}$
- g generiert $\mathbb{G}_1$, h generiert $\mathbb{G}_2$ und $e(g, h)$ generiert $\mathbb{G}_T$
- Es gibt effiziente Algorithmen, um Gruppenoperationen durchzuführen, die Abbildung zu berechnen, Gruppenelemente zu vergleichen und zu ermitteln, ob Elemente aus der Gruppe sind oder nicht.

Allgemein kann eine bilineare Abbildung folgendermaßen definiert werden [Cos17]:
Eine Abbildung $e : U \times V \to W$ mit $u, u_1, u_2 \in U$, $v, v_1, v_2 \in V$ sowie $a, b \in \mathbb{K}$ und U, V, W kommutative Gruppen heißt bilinear, falls sie folgende Eigenschaften erfüllt:

- $e(u_1 \cdot u_2, v) = e(u_1, v) \cdot e(u_2, v)$ und
- $e(u, v_1 \cdot v_2) = e(u, v_1) \cdot e(u, v_2)$

und somit gilt äquivalent

- $e(u^a, v) = e(u, v)^a$ und
- $e(u, v^b) = e(u, v)^b$

Für die AGHO Signatur sind die sogenannten *product-pairing equations*, später auch Verifikationsgleichungen genannt, von großer Bedeutung. Es sind Gleichungen, welche die folgende Form haben und die sehr hilfreiche Eigenschaft der Bilinearität besitzen:

$$\prod_i \prod_j e(A_i, B_j)^{a_{ij}} = Z \tag{3.12}$$

mit $A_1, A_2, ... \in \mathbb{G}_1$, $B_1, B_2, ... \in \mathbb{G}_2$, $Z \in \mathbb{G}_T$ und $a_{ij} \in \mathbb{Z}_R$ konstant [AGHO11].

Die AGHO Signatur besteht aus 3 Gruppenelementen (Signaturkomponenten) und wird durch zwei Verifikationsgleichungen überprüft. Für das Signaturverfahren sollen Nachrichten aus $\mathbb{G}_1^l$ signiert werden, wobei über eine bilineare Gruppe $(p, \mathbb{G}, \mathbb{H}, \mathbb{T}, e, g, h)$ gearbeitet wird [AGHO11].

- *Schlüsselerzeugung* [AGHO11]:
  Mit Hilfe des Tupels $(p, \mathbb{G}, \mathbb{H}, \mathbb{T}, e, g, h)$ wird ein asymmetrisches Schlüsselpaar (sk, vk) erzeugt mit:

$$\begin{aligned} sk &= (v, (w_i)_{i=1}^l, z) \\ vk &= (V, (W_i)_{i=1}^l, Z) = (h^v, (h^{w_i}), h^z) \end{aligned} \tag{3.13}$$

  mit $v, z, w_i \xleftarrow{\$} \mathbb{Z}_R$.
- *Signieren* [AGHO11]:
  Es sei $m = (m_i)_{i=1}^l$ aus $\mathbb{G}_1^l$ und $\sigma = Sig(sk, m) = (R, S, T) \in \mathbb{G}_1^2 \times \mathbb{G}_2$, $r \xleftarrow{\$} \mathbb{Z}_R$ und

$$\begin{aligned} R &= g^r \\ S &= g^{z-rv} \prod_i m_i^{-w_i} \\ T &= h^{\frac{1}{r}} \end{aligned} \tag{3.14}$$

  Zurückgegeben wird die Signatur (R, S, T).

- *Verifikation* [AGHO11]:
  Akzeptiere die Signatur, falls $R, S \in \mathbb{G}_1$ und $T \in \mathbb{G}_2$ und

$$e(R,V) \cdot e(S,h) \cdot \prod_i e(m_i, W_i) = e(g,Z) \tag{3.15}$$

  und

$$e(R,T) = e(g,h) \tag{3.16}$$

Die Korrektheit der Verifikationsgleichungen kann folgendermaßen gezeigt werden:
Für Gleichung 3.15 gilt folgendes:

$$\begin{aligned}
e(R,V) \cdot e(S,h) \cdot \prod_i e(m_i, W_i) &= e(g^r, V) \cdot e(g^{z-r\cdot v} \cdot \prod_i m_i^{-w_i}, h) \cdot \prod_i e(m_i, h^{w_i}) \\
&= e(g^r, V) e(g^{z-r\cdot v} \cdot \prod_i m_i^{-w_i}, h) \cdot \prod_i e(m_i, h)^{w_i} \\
&= e(g^r, V) \cdot e(g^{z-r\cdot v} \cdot \prod_i m_i^{-w_i}, h) \cdot e(\prod_i m_i^{w_i}, h) \\
&= e(g^r, V) \cdot e(g^{z-r\cdot v}, h) \cdot e(\prod_i m_i^{-w_i}, h) \cdot e(\prod_i m_i^{w_i}, h) \\
&= e(g^r, h^v) \cdot e(g^{z-r\cdot v}, h) \cdot e(0, h) \\
&= e(g^r, h)^v \cdot e(g^{z-r\cdot v}, h) \\
&= e(g^{r\cdot v}, h) \cdot e(g^{z-r\cdot v}, h) \\
&= e(g^{r\cdot v + z - r\cdot v}, h) \\
&= e(g^z, h) \\
&= e(g,h)^z \\
&= e(g, h^z) \\
&= e(g, Z)
\end{aligned}$$

somit ist die Gültigkeit gezeigt. Für Gleichung 3.16 gilt:

$$\begin{aligned}
e(R,T) &= e(g^r, h^{1/r}) \\
&= e(g, h^{1/r})^r \\
&= e(g,h)^{r\cdot 1/r} \\
&= e(g,h)
\end{aligned}$$

woraus die Korrektheit der beiden Verifikationsgleichungen folgt.

Das vorgestellte Signaturverfahren erfüllt viele Eigenschaften, die einem elektronischen Wahlsystem zugutekommen. Unter anderem gilt, dass die AGHO Signatur *strukturerhaltend* ist, was soviel bedeutet wie, dass sowohl die Nachrichten als auch die Schlüssel und die Signatur aus Elementen aus $\mathbb{G}_1$ und $\mathbb{G}_2$ bestehen.

3.4 Die restriktive AGHO Blindsignatur

Die restriktive Blindsignatur [HK19] basiert auf der soeben erläuterten AGHO Signatur, einem Commitment-Schema, zum Beispiel ein Verschlüsselungsverfahren, und zwei nicht-interaktiven Zero-Knowledge Proofs. Der Hauptunterschied besteht darin, dass der Unterzeichner /die Unterzeichnerin jetzt nicht mehr sehen kann, was unterzeichnet wird, aber

trotzdem einen Beweis darüber vorgelegt bekommt, dass es sich bei der vom User übertragenen Nachricht um eine gültige Nachricht handelt [HK19].

Der Ablauf des Verfahrens funktioniert folgendermaßen: Das Protokoll läuft zwischen einem Unterzeichner/einer Unterzeichnerin S und einem User/einer Userin U ab. Die Schlüsselerzeugung für die Signaturschlüssel und die Verifikationsschlüssel funktionieren wieder wie in Gleichung 3.13. Sei m die Nachricht, um die es sich handelt und auf die sich der User festlegen möchte. Um sich festzulegen berechnet er ein Commitment c auf m (später wird das die ElGamal Verschlüsselung der Nachricht sein). Anschließend verblindet U c mit Hilfe eines zufälligen Werts (*Pads*) P zu $\bar{c}$. Damit der Unterzeichner nicht über P auf c rückschließen kann, wird auch P durch eine einfache Exponentiation obfuskiert. Es wird durch einen nicht-interaktiven Zero-Knowledge Proof bewiesen, dass das verblindete Commitment $\bar{c}$ und das obfuskierte Pad, $\bar{P}$, die richtige Struktur/Form aufweisen. Bezüglich einer elektronischen Wahl könnte das bedeuten, dass der hinter dem Commitment stehende Stimmzettel zulässig ist. Das wäre zum Beispiel die Anforderung, dass ein gültiger Kandidat oder eine gültige Kandidatin gewählt wurde und gewisse Attribute des Wählers korrekt angeführt sind. Dieser Beweis ist notwendig, da S Informationen über das braucht, was unterzeichnet wird und nicht wahllos alles signiert, was ihm vorgelegt wird. Gleichzeitig wird aber trotzdem c und P vor S versteckt. Bei Gültigkeit des Beweises signiert S $\bar{c}$ und $\bar{P}$, wobei das Ergebnis noch keiner gültigen Signatur entspricht. Um eine gültige Signatur zu erhalten, werden die einzelnen Teile kombiniert. Diese Kombination entspricht dann nicht mehr einer Signatur auf $\bar{c}$, sondern ist eine Signatur auf das ursprüngliche Commitment c [HK19]. Die Nachricht m kann auch aus mehreren Teilen, später auch Attributen (z.B. Geschlecht, Alter,..) bestehen. Die folgenden Erklärungen beschränken sich für die Verständlichkeit auf ein Attribut m.

Verallgemeinert kann die restriktive AGHO Blindsignatur also folgendermaßen beschrieben werden: S hat den AGHO Signaturschlüssel $sk = (v, w, z)$, und U hat den passenden Verifikationsschlüssel vk. U hätte gerne ein Zertifikat (c, σ) auf das festgelegte m von S [HK19]:

1. U berechnet c auf m mit Opening w durch Verwenden vom Commitment-Verfahren Comm_{COM}. Im Bezug auf die Elektronischen Wahlen ist das eine ElGamal Verschlüsselung von m.
2. U verblindet c durch ein zufälliges Pad $P \xleftarrow{\$} \mathbb{G}_1$ und erhält $\bar{c} = (c \cdot P^{-1})$. Weiters wählt er/sie $e, f \xleftarrow{\$} \mathbb{Z}_R$ und eine zufällige Aufteilung $f = f_1 + f_2$ von f. U setzt $\bar{P} = P^e$ und $(G_1, G_2, G_3) = (g^e, g^{f_1}, g^{e \cdot f_2})$. Es handelt sich bei e also um einen geheimen Exponenten, um P zu verblinden. f ist ein Rauschfaktor, welcher über G von der Seite des Users eine Zufälligkeit in die Signatur bringt. Dies ist notwendig, um die Linkbarkeit zwischen dem von S gewählten Zufallswert (im folgenden r) und der Signatur zu vermeiden. Das heißt S kann später dadurch nicht mehr über r herausfinden, welche Nachricht er wann signiert hat. Dann sendet U $(\bar{c}, \bar{P}, (G_1, G_2, G_3))$ an S und führt einen NIZK über die korrekte Form durch:

$$\begin{aligned} \pi_U = \text{NIZK}[(\eta, \varphi_1, \varphi_2, \omega) : \\ G_1^{\eta} = g \wedge g^{\varphi_1} = G_2 \wedge G_1^{\varphi_2} = G_3 \wedge \\ \text{Vf}_{COM}(\bar{c} \cdot \bar{P}^{\eta}, x, \omega) = 1] \end{aligned} \tag{3.17}$$

   mit den Werten $(\eta, \varphi_1, \varphi_2, \omega) = (1/e, f_1, f_2, w)$, wobei Vf_{COM} die Verifikation der Gültigkeit des Stimmzettels beschreibt. Das bedeutet, dass U bezogen auf eine Wahl die Gültigkeit des Stimmzettels beweist, also dass die Nachricht tatsächlich hinter $\bar{c}$ steht.
3. S verifiziert π_U. Bei Gültigkeit generiert S eine zufällige Aufteilung $z = z_1 + z_2$ des

Signaturschlüssels z und berechnet die Signatur $\bar{\sigma} = (\bar{R}, \bar{S}_1, \bar{S}_2, \bar{T})$ mit $\bar{R} = g^r, \bar{T} = h^{1/r}, \bar{S}_1 = g^{z_1} \cdot G_2^{-r \cdot v} \cdot \bar{c}^{-w}, \bar{S}_2 = G_1^{z_2} \cdot G_3^{-r \cdot v} \cdot \bar{P}^{-w}$ wobei $r \xleftarrow{\$} \mathbb{Z}_R$. S gibt $\bar{\sigma}$ an U zurück. Zusätzlich erstellt S einen Beweis darüber, dass die Signatur die richtige Form hat:

$$\begin{aligned} \pi_S = \text{NIZK}[(\rho, \tau, \omega, \zeta_1, \zeta_2) : h^\omega = W \wedge h^{\zeta_1} \cdot h^{\zeta_2} = Z \wedge \\ g^\rho = \bar{R} \wedge \bar{T}^\rho = h \wedge V^\rho \cdot h^{-\tau} = 1 \wedge \\ g^{\zeta_1} \cdot G_2^{-\tau} \cdot \bar{m}^{-\omega} = \bar{S}_1 \wedge G_1^{\zeta_2} \cdot G_3^{-\tau} \cdot \bar{P}^{-\omega} = \bar{S}_2], \end{aligned} \tag{3.18}$$

mit den Werten $(\rho, \tau, \omega, \zeta_1, \zeta_2) = (r, r \cdot v, w, z_1, z_2)$.

4. U überprüft ob π_S gültig ist. Wenn ja, dann gibt er/sie c und $\sigma = (R, S, T)$ zurück, wobei $R = \bar{R}^f, S = \bar{S}_1 \cdot \bar{S}_2^{1/e}, T = \bar{T}^{1/f}$.

UserIn		**UnterzeichnerIn**
kennt (vk)		kennt (sk)
$P \xleftarrow{\$} \mathbb{G}_1$		
$e, f \xleftarrow{\$} \mathbb{Z}_q$		
$\bar{m} = m \cdot P^{-1}$		
$\bar{P} = P^e$		
$f = f_1 + f_2$ zufällige Zerlegung		
$G_1 = g^e, G_2 = g^{f_1}, G_3 = g^{e \cdot f_2}$		
$G = (G_1, G_2, G_3)$		
	$\xrightarrow{\bar{m}, \bar{P}, G, \pi_U}$	
		$r \xleftarrow{\$} \mathbb{Z}_q$
		$z = z_1 + z_2$ zufällige Zerlegung
		$\bar{S}_1 = g^{z_1} \cdot G_2^{-r \cdot v} \cdot \bar{m}^{-w}$
		$\bar{S}_2 = G_1^{z_2} \cdot G_3^{-r \cdot v} \cdot \bar{P}^{-w}$
		$\bar{R} = g^r, \bar{T} = h^{1/r}$
	$\xleftarrow{\bar{\sigma} = (\bar{R}, \bar{S}_1, \bar{S}_2, \bar{T}), \pi_S}$	
$S = \bar{S}_1 \cdot \bar{S}_2^{1/e}$		
$R = \bar{R}^f, T = \bar{T}^{1/f}$		

Abbildung 3.9: Der Ablauf der AGHO Blindsignatur nach [HK19]

In Abbildung 3.9 ist das Blindsignatur-Verfahren noch einmal graphisch dargestellt. Der Ablauf entspricht dem oben beschriebenen Verfahren. Für mehrere Attribute funktioniert der Ablauf des Signaturverfahrens analog.
Auch hier kann gezeigt werden, dass die Verifikationsgleichungen durch die Form der Signatur

erfüllt bleiben. Dafür wird zuerst das Tupel (R, S, T) ausgerechnet.

$$\begin{aligned}
R &= \bar{R}^f = g^{r \cdot f} \\
S &= \bar{S}_1 \cdot \bar{S}_2^{1/e} \\
&= g^{z_1} \cdot G_2^{-r \cdot v} \cdot \bar{c}^{-w} \cdot (G_1^{z_2} \cdot G_3^{-r \cdot v} \cdot \bar{P}^{-w})^{1/e} \\
&= g^{z_1} \cdot G_2^{-r \cdot v} \cdot \bar{c}^{-w} \cdot G_1^{z_2/e} \cdot G_3^{-r \cdot v/e} \cdot \bar{P}^{-w/e} \\
&= g^{z_1} \cdot g^{-r \cdot v \cdot f_1} \cdot \bar{c}^{-w} \cdot g^{z_2} \cdot g^{-r \cdot v \cdot f_2} \cdot \bar{P}^{-w/e} \\
&= g^{z_1 + z_2} \cdot g^{-r \cdot v \cdot (f_1 + f_2)} \cdot (\bar{c} \cdot P)^{-w} \\
&= g^{z - r \cdot v \cdot f} \cdot c^{-w} \\
T &= \bar{T}^{1/f} = h^{\frac{1}{r \cdot f}}
\end{aligned} \tag{3.19}$$

Wie schon bei der AGHO Signatur folgt die Gültigkeit der Verifikationsgleichungen folgendermaßen:

$$\begin{aligned}
e(R, V) \cdot e(S, h) \cdot \prod_i e(c_i, W_i) = & \\
&= e(g^{r \cdot f}, V) \cdot e(g^{z - r \cdot v \cdot f} \cdot c^{-w}, h) \cdot \prod_i e(c_i, W_i) \\
&= \cdots \text{(analog zur AGHO Signatur)} \\
&= e(g^{r \cdot v \cdot f}, h) \cdot e(g^{z - r \cdot v \cdot f}, h) \\
&= e(g^{r \cdot v \cdot f + z - r \cdot v \cdot f}, h) \\
&= e(g^z, h) \\
&= e(g, Z)
\end{aligned} \tag{3.20}$$

und

$$\begin{aligned}
e(R, T) &= e(g^{r \cdot f,}, h^{\frac{1}{r \cdot f}}) \\
&= e(g, h)^{\frac{r \cdot f}{r \cdot f}} \\
&= e(g, h)
\end{aligned} \tag{3.21}$$

4 Ein strukturerhaltendes elektronisches Wahlprotokoll

4.1 Encrypted Votes und der Protokollablauf

In diesem Abschnitt wird der Ablauf des Protokolls, welches auf der restriktiven AGHO Blindsignatur [HK19] beruht, beschrieben.
Für die Implementierung selbst werden einige Annahmen und Einschränkungen getroffen werden, da sonst der Rahmen dieser Masterarbeit gesprengt werden würde. Auf diese Einschränkungen wird in Abschnitt 4.4.5 genauer eingegangen.
Für den Protokollablauf wird eine Kommunikation zwischen drei Entitäten stattfinden:

- **Client** C, über den die WählerInnen ihre Stimme abgeben können
- **Wahlbehörde** W, welche verblindete Stimmen entgegennimmt und die AGHO Blindsignatur darauf ausstellen wird
- **Auszählbehörde** A, welche signierte, unverblindete Stimmen zur Auszählung entgegennimmt

Die Parameter des Wahlsystems bestehen aus einer bilinearen Gruppe $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ primer Ordnung R. Außerdem ist g ein Generator von $\mathbb{G}_1$ und h ein Generator von $\mathbb{G}_2$.
Das Wahlverfahren selbst besteht aus zwei wesentlichen Komponenten:

- die asymmetrische ElGamal Verschlüsselung auf $\mathbb{G}_1$
- die restriktive AGHO Blindsignatur (nach Abschnitt 3.4) von einer gewissen Anzahl an Elementen aus $\mathbb{G}_1$

4.1.1 Die Schlüsselerzeugung

Für die Generierung der sogenannten *Encrypted Vote* (der gesamte verschlüsselte Stimmzettel) wird eine ElGamal Verschlüsselung vorgenommen. Dafür wird ein Schlüsselpaar (sk_{EV}, pk_{EV}) benötigt, welches von der Auszählbehörde generiert wird. Dies kann auch ein Schlüsselpaar sein, welches in einer vorherigen Wahl schon verwendet wurde. Der zugehörige öffentliche Schlüssel pk_{EV} wird von A veröffentlicht, damit die Clients ihre Stimmen mit diesem verschlüsseln können. Das Schlüsselpaar wird folgendermaßen erzeugt:

$$\begin{aligned} sk_{EV} &\xleftarrow{\$} \mathbb{Z}_R \\ \underline{pk}_{EV} &= \underline{g}^{sk_{EV}} \end{aligned} \tag{4.1}$$

Hier und auch im Folgenden ist die Multiplikation (bzw. Punktaddition) zwischen zwei Punkten und auch die Potenz von Skalar und Punkt (bzw. Skalarmultiplikation) auf der elliptischen Kurve immer gemäß der Definition dieser Rechenoperationen für elliptische Kurven und der modularen Operationen zu verstehen. sk_{EV} liegt bei der Auszählbehörde und $\underline{pk}_{EV}$ wird von C verwendet, um die Stimme zu verschlüsseln.

Auch für die Durchführung der AGHO Blindsignatur (siehe Abschnitt 3.4) wird ein Schlüsselpaar (sk_σ, pk_σ) benötigt. Das Schlüsselpaar setzt sich folgendermaßen zusammen:

$$\begin{aligned} sk_\sigma &= (v, (w_i)_{i=1}^{2\cdot l}, z) \xleftarrow{\$} \mathbb{Z}_R \\ \underline{pk}_\sigma &= (\underline{h}^v, \cdot\underline{h}^{w_i}, \underline{h}^z) = (V, (W_i)_{i=1}^{2\cdot l}, Z) \end{aligned} \tag{4.2}$$

Aufgrund der Struktur der ElGamal Verschlüsselung (zwei Gruppenelemente Geheimtext pro Gruppenelement Klartext) werden hier doppelt so viele w's wie Attribute benötigt. Der private Schlüssel sk_σ liegt beim Wahlamt und $\underline{pk}_\sigma$ wird verwendet, um die Signatur zu verifizieren.

4.1.2 Die Signatur beim Wahlamt

Client		**Wahlamt**
kennt (pk_{EV}, pk_σ, c)		kennt (sk_σ)
$o \xleftarrow{\$} \mathbb{Z}_R$		
$c = (c_1, c_2) = (g^o, pk_{EV}^o \cdot m)$		
$P = (P_1, P_2) \xleftarrow{\$} \mathbb{G}_1^2$		
$\bar{c} = (\bar{c_1}, \bar{c_2}) =$		
$(c_1 \cdot P_1^{-1}, c_2 \cdot P_2^{-1})$		
$e, f \xleftarrow{\$} \mathbb{Z}_R, f = f_1 + f_2$		
$\bar{P} = (P_1^e, P_2^e)$		
$G = (G_1, G_2, G_3)$		
$= (g^e, g^{f_1}, g^{f_2 \cdot e})$		
	$\xrightarrow{\bar{c}, \bar{P}, G, \pi_U}$	
		Verify π_C,
		$r \xleftarrow{\$} \mathbb{Z}_R$
		$z = z_1 + z_2$
		$\bar{R} = g^r$
		$\bar{S}_1 = g^{z_1} \cdot G_2^{-r \cdot v} \cdot \bar{c}^{-w}$
		$\bar{S}_2 = G_1^{z_2} \cdot G_3^{-r \cdot v} \cdot \bar{P}^{-w}$
		$\bar{T} = h^{1/r}$
	$\xleftarrow{\bar{\sigma} = (\bar{R}, \bar{S}_1, \bar{S}_2, \bar{T}), \pi_S}$	
Verify π_W,		
$S = \bar{S}_1 \cdot \bar{S}_2^{1/e}$		
$R = \bar{R}^f, T = \bar{T}^{1/f}$		

Abbildung 4.1: Das Signaturverfahren bei Wahlamt mit nur einem Attribut

Ein Stimmzettel besteht aus der verschlüsselten Stimme und weiteren verschlüsselten Attributen. Die Attribute können identitätsbasierte Attribute sein, wie zum Beispiel das Alter,

das Geschlecht oder der Wahlsprengel. Sei diese Stimme, dargestellt als Punkt auf der elliptischen Kurve, bezeichnet mit $\underline{m} = (\underline{m_i})_{i=1}^{l}$ mit l als Anzahl der Attribute. Dann hat die Stimme folgende Form

$$\begin{pmatrix} \underline{m_1} \\ \underline{m_2} \\ \vdots \\ \underline{m_l} \end{pmatrix} \widehat{=} \begin{pmatrix} \text{Attribut 1} \\ \text{Attribut 2} \\ \vdots \\ \text{Auswahl/Stimme} \end{pmatrix}$$

Um die Stimme $\underline{m}$ zu verschlüsseln wird die ElGamal Verschlüsselung folgendermaßen durchgeführt:

$$\begin{aligned} &(o_i)_{i=1}^{l} \xleftarrow{\$} \mathbb{Z}_R \\ &(\underline{c}_i)_{i=1}^{l} = (\underline{c_1}, \underline{c_2})_i = (\underline{g}^{o_i}, \underline{pk}_{EV}^{o_i} \cdot \underline{m_i}) \end{aligned} \tag{4.3}$$

Die Encrypted Vote hat dann folgende Form:

$$\begin{pmatrix} c_1[1] & c_2[1] \\ c_1[2] & c_2[2] \\ \vdots & \vdots \\ c_1[l] & c_2[l] \end{pmatrix}$$

Im nächsten Schritt will C die Encrypted Vote an das Wahlamt schicken, um die verschlüsselte Stimme signieren zu lassen. Gemäß der AGHO Blindsignatur muss die Stimme davor verblindet werden. Dafür wird ein zufälliges Pad $(\underline{P}_i)_{i=1}^{l} = (\underline{P_1}, \underline{P_2})_i \xleftarrow{\$} \mathbb{G}_1^2$ gewählt. Die verblindete Stimme berechnet sich folgendermaßen:

$$(\bar{\underline{c}}_i)_{i=1}^{l} = (\bar{\underline{c_1}}, \bar{\underline{c_2}})_i = (\underline{c}_{1,i} \cdot \underline{P}_{1,i}^{-1}, \underline{c}_{2,i} \cdot \underline{P}_{2,i}^{-1}) \tag{4.4}$$

Sie hat die Form:

$$\begin{pmatrix} \bar{c}_1[1] & \bar{c}_2[1] \\ \bar{c}_1[2] & \bar{c}_2[2] \\ \vdots & \vdots \\ \bar{c}_1[l] & \bar{c}_2[l] \end{pmatrix}$$

Um auch das Pad zu veblinden sei $e, f \xleftarrow{\$} \mathbb{Z}_R$ mit einer zufälligen Zerlegung $f = f_1 + f_2$ und

$$(\bar{\underline{P}}_i)_{i=1}^{l} = (\underline{P}_{1,i}^{e}, \underline{P}_{2,i}^{e}) \tag{4.5}$$

Der Client berechnet außerdem $\underline{G} = (\underline{G_1}, \underline{G_2}, \underline{G_3}) = (\underline{g}^e, \underline{g}^{f_1}, \underline{g}^{e \cdot f_2})$. Dann sendet C an W folgendes: $(\bar{\underline{c}}, \bar{\underline{P}}, \underline{G}, \pi_C)$, wobei π_C ein nicht-interaktiver Zero-Knowledge Proof über die Korrektheit der Form des Stimmzettels (Stimme + Attribute) hinter dem verblindeten Ciphertext ist. Auf die Funktionsweise des Beweises wird in Abschnitt 4.1.5 genauer eingegangen.

Mit Hilfe der gegebenen Parameter kann W nun eine Signatur über die verblindete Stimme erstellen. Hierfür sei $r \xleftarrow{\$} \mathbb{Z}_R$ und $z = z_1 + z_2$ eine zufällige Zerlegung von z. Die Signatur

wird folgendermaßen berechnet:

$$\begin{aligned}
\bar{\underline{R}} &= \underline{g}^{r} \\
\bar{\underline{S}}_1 &= \underline{g}^{z_1} \cdot \underline{G}_2^{-r \cdot v} \cdot \prod_{i=1}^{l} \bar{\underline{c}}_i^{-w_i} \\
&= \underline{g}^{z_1} \cdot \underline{G}_2^{-r \cdot v} \cdot \prod_{i=1}^{l} (\bar{\underline{c}}_{1,i}^{-w_{1,i}} \cdot \bar{\underline{c}}_{2,i}^{-w_{2,i}}) \\
\bar{\underline{S}}_2 &= \underline{G}_1^{z_2} \cdot \underline{G}_3^{-r \cdot v} \cdot \prod_{i=1}^{l} \bar{\underline{P}}_i^{-w_i} \\
&= \underline{G}_1^{z_2} \cdot \underline{G}_3^{-r \cdot v} \cdot \prod_{i=1}^{l} (\bar{\underline{P}}_{1,i}^{-w_{1,i}} \cdot \bar{\underline{P}}_{2,i}^{-w_{2,i}}) \\
\bar{\underline{T}} &= \underline{h}^{1/r}
\end{aligned} \tag{4.6}$$

An C zurückgeschickt wird dann $(\bar{\sigma}, \pi_W) = ((\bar{\underline{R}}, \bar{\underline{S}}_1, \bar{\underline{S}}_2, \bar{\underline{T}}), \pi_W)$ wobei $\bar{\sigma}$ die Signatur bezüglich der verblindeten Stimme ist und π_W wieder ein nicht-interaktiver Zero-Knowledge Proof über die Korrektheit, diesmal seitens des Wahlamts, ist. Der soeben beschriebene Protokollablauf kann auch aus Abbildung 4.1 (exemplarisch für einen aus einem Attribut bestehenden Stimmzettel) entnommen werden.

4.1.3 Die Stimmabgabe bei der Auszählbehörde

Client		**Auszählbehörde**
kennt (pk_{EV}, pk_σ, c)		kennt (sk_{EV}, pk_σ)
$\sigma = (R, S, T)$		
	$\xrightarrow{c, \sigma}$	
		Verify:
		$e(R, V) \cdot e(S, h) \cdot e(c, W)$
		$\stackrel{?}{=} e(g, Z)$
		$e(R, T) \stackrel{?}{=} e(g, h)$

Abbildung 4.2: Stimmabgabe bei der Auszählbehörde

Um die Stimme schlussendlich zum Auszählen abzugeben (siehe Abbildung 4.2), wird die unverblindete Signatur σ, passend zur unverblindeten Encrypted Vote c, gebraucht. Dafür kann die Signatur folgendermaßen zu $\sigma = (R, S, T)$ zusammengesetzt werden:

$$\begin{aligned}
\underline{R} &= \bar{\underline{R}}^{f} \\
\underline{S} &= \bar{\underline{S}}_1 \cdot \bar{\underline{S}}_2^{1/e} \\
\underline{T} &= \bar{\underline{T}}^{1/f}
\end{aligned} \tag{4.7}$$

Die Encrypted Vote wird zusammen mit der Signatur (c, σ) *ohne Authentifizierung*, um die Anonymität zu gewährleisten, an die Auszählbehörde geschickt. Um sicher zu sein, dass die

Stimme gültig ist, werden die Verifikationsgleichungen überprüft:

$$\begin{aligned} e(\underline{R}, \underline{V}) \cdot e(\underline{S}, h) \cdot \prod_{i=1}^{2 \cdot l} e(\underline{c}_i, \underline{W}_i) &\stackrel{?}{=} e(\underline{g}, \underline{Z}) \\ e(\underline{R}, \underline{T}) &\stackrel{?}{=} e(\underline{g}, \underline{h}) \end{aligned} \tag{4.8}$$

A behält die Stimme bei Gültigkeit bis zur Auszählung verschlüsselt in der Datenbank.

4.1.4 Die Auszählung

Sobald die Auszählung beginnt, werden die Encrypted Votes aus der Datenbank geholt und mit dem privaten Schlüssel von A, sk_{EV}, entschlüsselt:

$$(\underline{m}_i)_{i=1}^{l} = \underline{c}_{2i} \cdot \underline{c}_{1i}^{-sk_{EV}} \tag{4.9}$$

Für die Ermittlung des Gewinners oder der Gewinnerin wird nur dass letzte Attribut von $m = (m_i)_{i=1}^{l}$ herangezogen, da sich dort die eigentliche Stimme befindet. Die übrigen Attribute sind für statistische Auswertungen interessant. Das Ergebnis wird anschließend von der Auszählbehörde veröffentlicht.

Zum Schluss muss noch ein Beweis über die Korrektheit der Entschlüsselung durchgeführt werden. Dafür wird ein Zero-Knowledge Proof erstellt, der zeigt, dass die entschlüsselten Stimmzettel tatsächlich mit dem zu pk_{EV} gehörendem sk_{EV} entschlüsselt wurden. Das bedeutet, dass für die Entschlüsselung der korrekte Schlüssel verwendet wurde.

4.1.5 Die Zero-Knowledge Proofs

Zero-Knowledge Proof über die korrekte Form des Stimmzettels

Für den Zero-Knowledge Proof, welchen der Client an das Wahlamt schickt, müssen nach Definition der AGHO Blindsignatur [HK19] vier Gleichungen erfüllt sein:

1. $G_1^{1/e} = g$
2. $g^{f_1} = G_2$
3. $G_1^{f_2} = G_3$
4. $Vf_{COM}(\bar{c} \cdot \bar{P}^{\eta}, m, w) = 1$

Für die erste Gleichung soll gezeigt werden, dass der Beweiser einen geheimen Exponenten $(1/e)$ kennt, sodass $G_1^{1/e} = G$. Für die zweite Gleichung, soll gezeigt werden, dass der Beweiser, C, einen geheimen Exponenten f_1 kennt, sodass die Gleichung $G^{f_1} = G_2$ erfüllt ist. Für die dritte Gleichung soll gezeigt werden, dass C einen geheimen Exponenten f_2 kennt, sodass $G_1^{f_2} = G_3$. Beim vierten Teil des Zero-Knowledge Proofs ist folgende Gleichung zu zeigen $Vf_{COM}(\bar{c} \cdot \bar{P}^{1/e}, m, w) = 1$, was so viel bedeutet wie: C kennt einen Wert $1/e$ (bzw. e), sodass $\bar{c} \cdot \bar{P}^{1/e} = \text{Enc}(pk_{EV}, m)$. Es soll aber außerdem gezeigt werden, dass der Ciphertext eine gültige Form hat, also, dass $(\bar{c_1}, \bar{c_2}) \cdot (\bar{P}_1^{1/e}, \bar{P}_2^{1/e}) = \text{Enc}(pk_{EV}, m) = (g^o, pk^o \cdot m)$ ist.
Die letzten zwei Gleichungen werden nur für öffentliche Attribute (wie zum Beispiel den Wahlsprengel, das Geschlecht oder das Alter, ...) durchgeführt, da die Stimme selbst dem Wahlamt verborgen bleiben soll. Nach der Beschreibung aus Abschnitt 4.1.2 werden also alle Attribute, außer dem letzten ($l - 1$ Attribute) herangezogen. Im Folgenden wird dieser

reduzierte Nachrichtenvektor mit $\hat{m} = (m_i)_{i=1}^{l-1}$ bezeichnet. Für die beiden zuletzt vorgestellten Gleichungen gilt außerdem:

$$\begin{aligned} \bar{c_1} \cdot \bar{P}_1^{1/e} = g^o \Leftrightarrow g^o \cdot \bar{P}_1^{-1/e} = \bar{c_1} \\ \bar{c_2} \cdot \bar{P}_2^{1/e} = pk_{EV}^o \cdot \hat{m} \Leftrightarrow pk_{EV}^o \cdot \bar{P}_2^{-1/e} = \bar{c}_2 \cdot \hat{m}^{-1} \end{aligned} \tag{4.10}$$

bzw. für mehrere Attribute ausgeschrieben:

$$\begin{aligned} g^{o_i} \cdot \bar{P}_{1,i}^{-1/e} = \bar{c}_{1,i} \text{ für } i = 1, ..., l \\ pk_{EV}^{o_i} \cdot \bar{P}_{2,i}^{-1/e} = \bar{c}_{2,i} \cdot m_i^{-1} \text{ für } i = 1, ..., l-1 \end{aligned} \tag{4.11}$$

Folgendermaßen ergibt sich daraus der Ablauf des Beweises:

- Sei $\nu \xleftarrow{\$} \mathbb{Z}_R^{l+3}$ mit

$$\nu = \begin{pmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \\ \vec{\nu_4} \end{pmatrix} \tag{4.12}$$

  wobei $\vec{\nu_4} \in \mathbb{Z}_R^l$ hier für den Vektor an Zufallszahlen für die o_is steht.
- Dann wird N berechnet durch:

$$N = \begin{pmatrix} N_1 \\ N_2 \\ N_3 \\ \vec{N_4} \\ \vec{N_5} \end{pmatrix} = \begin{pmatrix} G_1^{\nu_1} \\ g^{\nu_2} \\ G_1^{\nu_3} \\ g^{\vec{\nu_4}} \cdot \bar{P}_1^{-\nu_1} \\ pk_{EV}^{\vec{\nu_4}} \cdot \bar{P}_2^{-\nu_1} \end{pmatrix} \tag{4.13}$$

  wobei $g^{\vec{\nu_4}} \cdot \bar{P}_1^{-\nu_1} = g^{\nu_{4_i}} \cdot \bar{P}_{1,i}^{-\nu_1}$ für $i = 1, ..., l$ und $pk_{EV}^{\vec{\nu_4}} \cdot \bar{P}_2^{-\nu_1} = pk_{EV}^{\nu_{4_i}} \cdot \bar{P}_{2,i}^{-\nu_1}$ für $i = 1, ...l-1$
- Die Challenge berechnet sich aus $ch = H(N)$ wobei:

$$\begin{aligned} H(N) &= H(N_1||N_2||N_3||\vec{N_4}||\vec{N_5}) \\ &= H(N_1||N_2||N_3||N_{4_1}||...||N_{4_l}||N_{5_1}||...||N_{5_{l-1}}) \end{aligned} \tag{4.14}$$

- Die Response r ist dann:

$$\begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ \vec{r_4} \end{pmatrix} = ch \cdot \begin{pmatrix} (1/e) \\ f_1 \\ f_2 \\ o = (o_i)_{i=1}^l \end{pmatrix} + \begin{pmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \\ \vec{\nu_4} \end{pmatrix} \tag{4.15}$$

- Als Ausgabe erhält der Client $\pi_C = (ch, (r_1, r_2, r_3, \vec{r_4}))$, welche er an das Wahlamt schickt.
- Um die Verifikation durchführen zu können, müssen ausschließlich öffentliche (dem Wahlamt bekannte) Parameter verwendet werden. Um das zu erreichen können fol-

gende Umformungen gemacht werden:

$$\begin{aligned}
G_1^{r_1} &= G_1^{ch\cdot(1/e)+\nu_1} = g^{ch} \cdot N_1 \\
g^{r_2} &= g^{ch\cdot(f_1)+\nu_2} = G_2^{ch} \cdot N_2 \\
G_1^{r_3} &= G_1^{ch\cdot(f_2)+\nu_3} = G_3^{ch} \cdot N_3 \\
g^{\vec{r_4}} \cdot \bar{P}_1^{-r_1} &= g^{ch\cdot o+\vec{\nu_4}} \cdot \bar{P}_1^{-(ch\cdot 1/e+\nu_1)} \\
&= (g^o)^{ch} \cdot g^{\vec{\nu_4}} \cdot (\bar{P}_1^{-1/e})^{ch} \cdot \bar{P}_1^{-\nu_1} \\
&= (g^o \cdot \bar{P}_1^{-1/e})^{ch} \cdot g^{\vec{\nu_4}} \cdot \bar{P}_1^{-\nu_1} \\
&= \bar{c}_1^{ch} \cdot \vec{N}_4 \\
pk_{EV}^{\vec{r_4}} \cdot \bar{P}_2^{-r_1} &= pk_{EV}^{ch\cdot o+\vec{\nu_4}} \cdot \bar{P}_2^{-(ch\cdot 1/e+\nu_1)} \\
&= (pk_{EV}^o \cdot \bar{P}_2^{-1/e})^{ch} \cdot pk_{EV}^{\vec{\nu_4}} \cdot \bar{P}_2^{-\nu_1} \\
&= (\bar{c}_2 \cdot \hat{m}^{-1})^{ch} \cdot \vec{N}_5
\end{aligned} \tag{4.16}$$

wobei o, P_1, P_2 und c_1, c_2 als Vektoren der einzelnen Attribute zu verstehen sind. Also folgt insgesamt:

$$\begin{aligned}
N_1 &= G_1^{r_1} \cdot g^{-ch} \\
N_2 &= g^{r_2} \cdot G_2^{-ch} \\
N_3 &= G_1^{r_3} \cdot G_3^{-ch} \\
\vec{N}_4 &= g^{\vec{r_4}} \cdot \bar{P}_1^{-r_1} \cdot \bar{c}_1^{-ch} \\
\vec{N}_5 &= pk_{EV}^{\vec{r_4}} \cdot \bar{P}_2^{-r_1} \cdot (\bar{c}_2 \cdot \hat{m}^{-1})^{-ch}
\end{aligned} \tag{4.17}$$

Für die Verifikation müssen also nur noch die in 4.17 errechneten Terme für die N's in die folgende Gleichung eingesetzt werden, um zu überprüfen, ob das Ergebnis mit der ursprünglichen Challenge übereinstimmt.

$$ch \stackrel{?}{=} H(N_1||N_2||N_3||\vec{N_4}||\vec{N_5}) \tag{4.18}$$

Zero-Knowledge Proof über das korrekte Verhalten von W

Auch hier müssen nach Definition der AGHO Blindsignatur [HK19] die folgenden Gleichheiten durch einen Zero-Knowledge Proof gezeigt werden:

1. $h^{w_i} = W_i$
2. $h^{z_1} \cdot h^{z_2} = Z$
3. $g^r = \bar{R}$
4. $\bar{T}^r = h$
5. $V^r \cdot h^{-(r\cdot v)} = 1$
6. $g^{z_1} \cdot G_2^{-r\cdot v} \cdot \bar{c}^{-w} \hat{=} g^{z_1} \cdot G_2^{-r\cdot v} \cdot \prod_{i=1}^{l} (\bar{c}_{1,i}^{-w_{1,i}} \cdot \bar{c}_{2,i}^{-w_{2,i}}) = \bar{S}_1$
7. $G_1^{z_2} \cdot G_3^{-r\cdot v} \cdot \prod_{i=1}^{l} (\bar{P}_{1,i}^{-w_{1,i}} \cdot \bar{P}_{2,i}^{-w_{2,i}}) = \bar{S}_2$

Wie schon für die Client-Seite muss wieder ein mehrdimensionaler Zero-Knowledge Proof durchgeführt werden. Es soll dieser für die geheimen Exponenten $((w_i)_{i=1}^{2\cdot l}, z_1, z_2, r, -r \cdot v)$ durchgeführt werden. Der Ablauf des Zero-Knowledge Proofs ist damit folgender:

- Sei $\nu = \begin{pmatrix} \vec{\nu_1} \\ \nu_2 \\ \nu_3 \\ \nu_4 \\ \nu_5 \end{pmatrix} \xleftarrow{\$} \mathbb{Z}_R^{2 \cdot l+4}$
  wobei $\vec{\nu_1} \in \mathbb{Z}_R^{2 \cdot l}$ hier für den Vektor an Zufallszahlen für die w_i's steht.
- N berechnet sich folgendermaßen:

$$\begin{pmatrix} \vec{N_1} \\ N_2 \\ N_3 \\ N_4 \\ N_5 \\ N_6 \\ N_7 \end{pmatrix} = \begin{pmatrix} h^{\vec{\nu_1}} \\ h^{\nu_2} \cdot h^{\nu_3} \\ g^{\nu_4} \\ \bar{T}^{\nu_4} \\ V^{\nu_4} \cdot h^{\nu_5} \\ g^{\nu_2} \cdot G_2^{\nu_5} \cdot \bar{c}^{-\vec{\nu_1}} \\ G_1^{\nu_3} \cdot G_3^{\nu_5} \cdot \bar{P}^{-\vec{\nu_1}} \end{pmatrix} \tag{4.19}$$

  wobei folgendes gilt: $\bar{c}^{-\vec{\nu_1}} = \prod_{i=1}^{l} (\bar{c}_{1,i}^{-\nu_{1_{1,i}}} \cdot \bar{c}_{2,i}^{-\nu_{1_{2,i}}})$ und $\bar{P}^{-\vec{\nu_1}} = \prod_{i=1}^{l} (\bar{P}_{1,i}^{-\nu_{1_{1,i}}} \cdot \bar{P}_{2,i}^{-\nu_{1_{2,i}}})$.
- Daher wird die Challenge generiert durch:

$$\begin{aligned} ch &= H(N) \\ &= H(\vec{N_1}||N_2||N_3||N_4||N_5||N_6||N_7) \\ &= H(N_{1_1}||...||N_{1_{2 \cdot l}}||N_2||N_3||N_4||N_5||N_6||N_7) \end{aligned} \tag{4.20}$$

- Die Response r wird berechnet durch:

$$\begin{pmatrix} \vec{r_1} \\ r_2 \\ r_3 \\ r_4 \\ r_5 \end{pmatrix} = ch \cdot \begin{pmatrix} w = (w_i)_{i=1}^{2 \cdot l} \\ z_1 \\ z_2 \\ r \\ -r \cdot v \end{pmatrix} + \begin{pmatrix} \vec{\nu_1} \\ \nu_2 \\ \nu_3 \\ \nu_4 \\ \nu_5 \end{pmatrix} \tag{4.21}$$

- Das Wahlamt bekommt $\pi_W = (ch, (\vec{r_1}, r_2, r_3, r_4, r_5))$ als Ausgabe und schickt dies an den Client zurück.
- Für die Herleitung der zu verwendenden Parameter für die Verifikation gilt folgendes:

$$\begin{aligned} h^{\vec{r_1}} = h^{ch \cdot w + \vec{\nu_1}} &= W^{ch} \cdot \vec{N_1} \\ h^{r_2} \cdot h^{r_3} = h^{ch \cdot z_1 + \nu_2} \cdot h^{ch \cdot z_2 + \nu_3} &= (h^{z_1 + z_2})^{ch} \cdot N_2 \\ g^{r_4} = g^{ch \cdot r + \nu_4} &= \bar{R}^{ch} \cdot N_3 \\ \bar{T}^{r_4} = \bar{T}^{ch \cdot r + \nu_4} &= h^{ch} \cdot N_4 \\ V^{r_4} \cdot h^{r_5} = V^{ch \cdot r + \nu_4} \cdot h^{ch \cdot (-r \cdot v) + \nu_5} &= 1^{ch} \cdot N_5 \\ g^{r_2} \cdot G_2^{r_5} \cdot \bar{c}^{-\vec{r_1}} = g^{ch \cdot z_1 + \nu_2} \cdot G_2^{ch \cdot (-r \cdot v) + \nu_5} \cdot \bar{c}^{-(ch \cdot w + \vec{\nu_1})} &= \bar{S}_1^{ch} \cdot N_6 \\ G_1^{r_3} \cdot G_3^{r_5} \cdot \bar{P}^{-\vec{r_1}} = G_1^{ch \cdot z_2 + \nu_3} \cdot G_3^{ch \cdot (-r \cdot v) + \nu_5} \cdot \bar{P}^{-(ch \cdot w + \vec{\nu_1})} &= \bar{S}_2^{ch} \cdot N_7 \end{aligned} \tag{4.22}$$

  wobei auch hier bei den letzten zwei Gleichungen $\bar{c}^{-(ch \cdot w + \vec{\nu_1})}$ und $\bar{P}^{-(ch \cdot w + \vec{\nu_1})}$ als Produkt der einzelnen Komponenten und w und W als Vektor der Attribute zu verstehen sind.

Es gilt also:

$$\begin{aligned}
\vec{N_1} &= h^{\vec{r_1}} \cdot W^{-ch} \\
N_2 &= h^{r_2+r_3} \cdot Z^{-ch} \\
N_3 &= g^{r_4} \cdot \bar{R}^{-ch} \\
N_4 &= \bar{T}^{r_4} \cdot h^{-ch} \\
N_5 &= V^{r_4} \cdot h^{r_5} \cdot 1^{-ch} = V^{r_4} \cdot h^{r_5} \\
N_6 &= g^{r_2} \cdot G_2^{r_5} \cdot \bar{c}^{-\vec{r_1}} \cdot \bar{S}_1^{-ch} \\
N_7 &= G_1^{r_3} \cdot G_3^{r_5} \cdot \bar{P}^{-\vec{r_1}} \cdot \bar{S_2}^{-ch}
\end{aligned} \tag{4.23}$$

Somit kann die Gültigkeit auch hier durch einfaches Einsetzen der hergeleiteten Parameter für die Ns überprüft werden:

$$ch \stackrel{?}{=} H(\vec{N_1}||N_2||N_3||N_4||N_5||N_6||N_7) \tag{4.24}$$

Zero-Knowledge Proof über die korrekte Entschlüsselung der Stimmen von A

Es soll hier gezeigt werden, dass die Encrypted Votes korrekt entschlüsselt wurden. Das bedeutet es ist zu zeigen, dass der Schlüssel sk mit dem die Stimmzettel entschlüsselt wurden, tatsächlich der zu pk_{EV} gehörende Schlüssel sk_{EV} ist. Dafür wird gezeigt, dass m tatsächlich die Entschlüsselung von (c_1, c_2) ist und dass pk_{EV} aus sk errechnet werden kann, also, dass:

$$\begin{aligned}
c_1^{sk_{EV}} &= c_2 \cdot m^{-1} \\
g^{sk_{EV}} &= pk_{EV}
\end{aligned} \tag{4.25}$$

Die Aussage lautet daher: *Ich kenne ein sk, sodass die Gleichungen aus 4.25 erfüllt sind.* Hierfür wird folgendermaßen vorgegangen:

- Sei $\nu \stackrel{\$}{\leftarrow} \mathbb{Z}_R$
- N wird berechnet durch:
$$N = \begin{pmatrix} N_1 \\ N_2 \end{pmatrix} = \begin{pmatrix} c_1^{\nu} \\ g^{\nu} \end{pmatrix} \tag{4.26}$$
- Die Challenge ergibt sich daher aus $ch = H(N_1||N_2)$
- Die Response ist $r = ch \cdot (sk_{EV}) + \nu$
- Die Ausgabe der Auszählbehörde ist $\pi_A = (ch, r)$ und wird zusammen mit den entschlüsselten Stimmen vom der Auszählbehörde veröffentlicht
- Die Verifikation läuft folgendermaßen ab:

$$\begin{aligned}
c_1^r &= c_1^{ch \cdot sk_{EV} + \nu} = (c_2 \cdot m^{-1})^{ch} \cdot N_1 \\
g^r &= g^{ch \cdot sk_{EV} \cdot \nu} = pk^{ch} \cdot N_2 \\
ch &\stackrel{?}{=} H((c_1^r \cdot (c_2 \cdot m^{-1})^{-ch})||(g^r \cdot pk^{-ch}))
\end{aligned} \tag{4.27}$$

4.2 Diskussion der IT-Sicherheitsbedingungen

Im folgenden Abschnitt werden die Bedingungen aus Abschnitt 2.1 für das soeben beschriebene Protokoll behandelt.

Verfügbarkeit

Da die Verfügbarkeit hauptsächlich von der Umsetzung der Ressourcen abhängt kann hier keine Aussage über die Erfüllung der Bedingung getroffen werden.
Wenn dieses Wahlsystem für eine Wahl eingesetzt wird, so wird eine Form von Replikation benötigt, um die Verfügbarkeit so weit wie möglich zu gewährleisten.

Anonymität

Für diese Bedingung müssen die Attribute, sowie auch die Entitäten (das Wahlamt und die Auszählbehörde) betrachtet werden. Da es sich beim Signieren um eine Blindsignatur handelt, weiß die Wahlbehörde nicht, welche Stimmauswahl sie gerade signiert. Es wird nur überprüft, ob die Stimme, welche sich unter der verblindeten Encrypted Vote versteckt, die korrekte Form hat. Somit kann das Wahlamt, sollte es diese Stimme später unverblindet zu Gesicht bekommen, nicht mehr feststellen, zu welcher Sitzung diese Stimme gehört. Es kann nur noch sehen, dass es diese Stimme offenbar signiert hat.
Eine Anwendung, welche die Nützlichkeit dieser Eigenschaft zeigt ist die Veröffentlichung der Encrypted Votes zusammen mit den unverschlüsselten Stimmen am Web Bulletin Board, um die Verifizierbarkeit zuzulassen.
Um seitens der Auszählbehörde keine Rückschlüsse von der Stimme auf die Wählerin zu erlauben, könnte die Stimmabgabe ohne Authentifizierung dieser stattfinden. Dies schützt die Anonymität, da sich in der Stimme keine komplett de-anonymisierenden Daten befinden. Diese Stimmabgabe sollte aber trotzdem über einen Onion Service, wie zum Beispiel *Tor*, durchgeführt werden, um durch die IP-Adresse nicht auf die Person rückschließen zu können.

Wähler Nachweisbarkeit

Sollte am Web Bulletin Board sowohl die Encryped Votes als auch die unverschlüsselten Stimmen veröffentlicht werden, wäre es möglich, dass ein Wähler seine Stimme dort wieder finden kann. Da die Eindeutigkeit der Stimmen aber nicht gegeben sein muss, muss für die Erfüllung der Bedingung ein zusätzlicher, eindeutiger Wert mit jeder Stimme veröffentlicht werden, den der passende Wähler selbst ebenfalls hat. Dann kann jeder Wähler überprüfen, ob seine Stimme korrekt gezählt wurde. Dies ist durch das Verblinden der Stimmen möglich, ohne die Anonymität gegenüber dem Wahlamt zu gefährden. Sollte ein Wähler seine Stimme auf dem Board nicht vorfinden, so kann ein Mechanismus eingebaut werden, der es zulässt, dass sich dieser Wähler beim Wahlamt beschwert und eine Neuauszählung ausgelöst wird.

Universelle Nachweisbarkeit

Wie bei der *Wähler-Nachweisbarkeit* schon beschrieben wurde, können auf dem Web Bulletin Board nach der Wahl sowohl die Encrypted Votes als auch die Stimmen im Klartext veröffentlicht werden. Das bedeutet, dass jede Person, welche die Auszählung überprüfen /verifizieren möchte, das auch tun kann.

Integrität

Um die Datenintegrität zu betrachten, muss der Datenaustausch zwischen den einzelnen Parteien betrachtet werden. Wenn zwischen Wähler und Wahlamt kommuniziert wird, ist die Integrität aus folgendem Grund gegeben: Der Wähler sendet seine verblindete Stimme an W.

Sollte die Stimme auf diesem Weg manipuliert werden, so wird beim Entblinden nicht mehr die ursprüngliche Stimme zurückgegeben werden und die Manipulation wird erkannt. Sollte die Nachricht erst am Rückweg manipuliert werden, so wird entweder die Stimme oder aber die Signatur nicht mehr korrekt sein.
Auch bei der Kommunikation zwischen dem Wähler und der Auszählbehörde lässt sich eine Manipulation leicht erkennen. Entweder die Stimme oder aber die Signatur werden nicht mehr gültig sein.

Wählbarkeit

Für die Wählbarkeit müssen zwei Bedingungen überprüft werden. Es muss sichergestellt werden, dass nur wahlberechtigte Personen wählen dürfen und dass Personen, welche schon gewählt haben, kein zweites Mal wählen können. Beide Faktoren dieser Bedingung hängen nicht vom Protokollaufbau, sondern von der Implementierung ab.
Die erste Bedingung könnte bei einer Implementierung durch eine einfache Datenbankabfrage überprüft werden, wenn jede Wähler-Entität einen Eintrag hat, welcher angibt, ob die Person wahlberechtigt ist. Sollte dies nicht der Fall sein und die Person trotzdem eine Stimme abgeben wollen, so kann schon das Ausstellen einer Signatur seitens des Wahlamts verweigert werden.
Für das Erfüllen der zweiten Bedingung wird ein Mechanismus gebraucht, der verhindert, dass ein Client zwei Mal dieselbe Signatur an die Auszählbehörde schickt, sollte seitens der Auszählbehörde keine Authentifizierung eingebaut werden.

Keine Belegbarkeit

Dieses Protokoll legt den Fokus auf die Möglichkeit, dass durch die identitätsbasierten Faktoren statistische Auswertungen der Wählerschaft durchgeführt werden können. Sollten die Encrypted Votes zusammen mit den unverschlüsselten Stimmen am Web Bulletin Board veröffentlicht werden, kann die Wählerin ganz einfach ihre Stimme auf dem Board suchen. Somit wäre diese Bedingung nicht erfüllt.

Robustheit

Durch die Form der restriktiven AGHO Blindsignatur ist diese Bedingung erfüllt. Die Stimme wird nur bei Gültigkeit (erfolgreicher Zero-Knowledge Proof seitens der Wählerin) signiert. Die Encrypted Vote muss also die richtige Form haben, damit ein Zertifikat auf die Stimme ausgestellt wird. Dazu zählt auch, dass die korrekte Anzahl an Personen gewählt wurde. Diese Überprüfung passiert aufgrund des Zero-Knowledge Proofs ohne, dass das Wahlamt erfährt, wie die Stimme tatsächlich aussieht.

Fairness

Sollten sowohl die verschlüsselten als auch die entschlüsselten Stimmen erst nach der Wahl (während der Auszählung) am Web Bulletin Board veröffentlicht werden, können von den bereits abgegebenen Stimmen keine Teilergebnisse von anderen Personen errechnet werden.
Von der Auszählungsbehörde selbst sollten während der Auszählung auch keine Teilergebnisse veröffentlicht werden (Schwellwert-Kryptosystem). Die Auszählung sollte erst dann begonnen werden, wenn die Wahl abgeschlossen ist. Wenn alle genannten Faktoren berücksichtigt werden, ist diese Bedingung erfüllt.

Korrektheit

Um die Korrektheit der Wahl zu gewährleisten, wird am Ende der Auszählung ein Beweis über die korrekte Auszählung der Stimmen erstellt. Der einfache Weg würde darin liegen, den privaten Schlüssel der Wahl zu veröffentlichen, damit jeder nachprüfen kann, ob die Stimmen korrekt entschlüsselt wurden, und damit die Auszählung auch korrekt stattgefunden hat. Da ein Schlüsselpaar aber auch öfter verwendet werden könnte und dieser Ansatz nicht sehr schön ist, wird für den Beweis ein nicht-interaktiver Zero-Knowledge Proof herangezogen. Durch diesen soll gezeigt werden, dass die entschlüsselten Stimmen tatsächlich mit dem zu pk_{EV} gehörendem sk_{EV} entschlüsselt wurden. Das bedeutet, dass für die Entschlüsselung der korrekte Schlüssel verwendet wurde.

Zwang zur Wahl

Wie schon in Abschnitt 2.1 besprochen, ist diese Bedingung, vor allem für Remote Systeme, besonders schwer zu erfüllen. Es muss die Möglichkeit unterbunden werden, dass eine Person zu einer bestimmten Auswahl gezwungen wird. Zwar könnte als erster Schritt die Möglichkeit eingebaut werden, dass eine Person ihre Stimme wieder annullieren lassen kann, trotzdem ist damit die Bedingung noch nicht erfüllt. Ein Erpresser könnte die abgegebene Stimme am Schluss auf dem Web Bulletin Board suchen und feststellen, dass die Stimme nicht gezählt wurde oder nicht aufscheint, sollten diese veröffentlicht werden.

4.3 Implementierung des Protokolls als kryptographische Bibliothek

In diesem Abschnitt wird auf die konkrete Umsetzung des in Abschnitt 4.1 erläuterten Protokolls eingegangen und diese beschrieben. Die Implementierung ist auf Anfrage auch unter GitHub zur Einsicht verfügbar [Kap20].
Die Implementierung des Protokolls wurde als kryptographische Bibliothek umgesetzt und wurde in der Skriptsprache *Python3* verfasst. Für die Realisierung von elliptischen Kurven und Pairings wurde die Bibliothek *Charm* [AMG+11], welche auf PBC [Lyn] basiert, verwendet. Charm erlaubt das Arbeiten und Ausführen von Operationen mit Pairings und stellt einen Teil der benötigten Implementierung für die ElGamal Verschlüsselung zu Verfügung. Für die Implementierung der Bibliothek (des Protokolls) wurde in fünf Schritten vorgegangen:

1. Implementierung der ElGamal Verschlüsselung
2. Implementierung der AGHO Signatur
3. Implementierung der AGHO Blindsignatur
4. Implementierung der ZKPs
5. Testen aller genannten Komponenten

4.3.1 Die Charm Bibliothek

Um die Implementierung des Protokolls durchführen zu können wurde im ersten Schritt die gewählte Bibliothek analysiert. Folgende Funktionalitäten, welche für die Implementierung der kryptographischen Bibliothek des AGHO Protokolls wichtig waren, werden durch Charm implementiert:

- Zur Verfügung stellen eines Typ-3 Pairings $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ zu Sicherheitsmaßzahl 128 Bit, wobei für $\mathbb{G}_1$ die Pairing-freundliche Bareto-Naehrig Kurve *BN254* [YCK+19] gewählt wurde.
- Erstellen von zufälligen Werten aus den Gruppen $\mathbb{G}_1, \mathbb{G}_2$ und $\mathbb{Z}_R$
- Die benötigten Rechenoperationen gemäß der EC Kryptographie
- Abbilden eines Integer-Wertes oder eines Strings in die Gruppen $\mathbb{G}_1$, $\mathbb{G}_2$ oder auf $\mathbb{Z}_R$ (als Datentyp *pairing.Element*) durch Hashen.
- Serialisieren von Elementen des Datentyps *pairing.Element* und anschließendes Deserialisieren dieser Elemente (Darstellung im base64 Format)
- Bilden des Pairing-Produkts $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ zweier Elemente g, h mit $g \in \mathbb{G}_1$, $h \in \mathbb{G}_2$ und $e(g, h) \in \mathbb{G}_T$

Trotz dem, dass diese Bibliothek nicht mehr regelmäßig gewartet wird, nur ein veraltetes Angebot an unterstützten Pairing-freundlichen Kurven bietet (es ist nur eine einzige Kurve zu 128 Bit Sicherheit vorhanden, alle anderen Kurven liegen darunter) und auch einen kleinen Fehler in der Implementierung beinhaltet (siehe Abschnitt 4.3.6), wird sie verwendet, da sie eine von sehr wenigen Bibliotheken mit Pairing Implementierung ist, welche alle benötigten Anforderungen erfüllt und eine sehr einfache Anbindung zu Python bietet.

4.3.2 Die ElGamal Verschlüsselung

Da Charm schon einige Komponententests für die Funktionalität der EC Operationen implementiert hat, konnte ein Teil der ElGamal Verschlüsselung für diese Klasse übernommen werden. Es wurde hier mit zwei Objekten gearbeitet. Die *ElGamal* Klasse, welche die Verschlüsselung von Nachrichten vornehmen soll und die *ElGamalCipher* Klasse, welche einen ElGamal Ciphertext definiert.
Dieses Script soll als Hilfestellung für die ElGamal Verschlüsselung gelten. Folgende Funktionalitäten sind enthalten:

- Schlüsselgenerierung eines ElGamal Schlüsselpaares
- Verschlüsseln eines Klartextes
- Entschlüsselung eines ElGamal Ciphertextes
- Erstellen eines NIZKs über die korrekte Entschlüsselung der Stimmen (nach Abschnitt 4.1.5)
- Verifikation eines NIZKs über die korrekte Entschlüsselung der Stimme

Eine wichtige Anmerkung ist, dass es bei der Entschlüsselung der Stimmen notwendig war eine Art Lookup-Tabelle der möglichen Klartexte bereitzustellen. Dies ist notwendig, da die Klartexte zuerst mit Hilfe der von Charm zu Verfügung gestellten Funktion *pairing.hash()* auf die Kurve gehasht werden (und die Nachrichten dafür encodiert werden) und somit auch wieder decodiert werden müssen. Dies bietet allerdings keine Einschränkung, da die möglichen Klartexte ohnehin bekannt sind.

4.3.3 Die AGHO Signatur

Um die restriktive AGHO Blindsignatur [HK19] zu implementieren wurde zuerst die klassische AGHO Signatur [AGHO11] implementiert. Wie in Abschnitt 3.4 beschrieben, sind folgende Funktionalitäten in diesem Skript enthalten:

- Erstellen eines Signatur Schlüsselpaares

- Signieren einer ElGamal verschlüsselten Nachricht
- Verifikation einer Signatur

Die Verifikation der Signatur wurde gemäß den Gleichungen aus 4.1.3 durchgeführt.

4.3.4 Die AGHO Blindsignatur

Zusätzlich zu den für die AGHO Signatur [AGHO11] benötigten Funktionalitäten wurden einige Funktionen ergänzt, um die restriktive AGHO Blindsignatur [HK19] zu erhalten. Insgesamt sind folgende Funktionen in dem resultierenden Skript enthalten:

- Erstellen eines Signatur Schlüsselpaares
- Verblinden eines ElGamal Ciphertextes für das Signaturverfahren (siehe Listing 4.1)
- Signieren einer verblindeten, ElGamal verschlüsselten Nachricht (siehe Listing 4.2)
- Entblinden der vom Server erhaltenen Signatur
- Verifikation einer AGHO Blindsignatur
- Erstellen eines nicht-interaktiven ZKPs vom Client an das Wahlamt über die Gültigkeit der Stimme nach Abschnitt 4.1.5
- Erstellen eines nicht-interaktiven ZKPs vom Wahlamt an den Client über die Gültigkeit der Signatur nach Abschnitt 4.1.5
- Verifikation des NIZKs vom Client
- Verifikation des NIZKs vom Wahlamt

Wie auch schon in Abschnitt 3.4 beschrieben liegt der wichtige Unterschied zwischen dem klassischen AGHO Signaturverfahren [AGHO11] und der restriktiven AGHO Blindsignatur [HK19] darin, dass der Server nicht mehr sehen kann, was er signiert, weil die Nachricht verblindet wird. Aus Listing 4.1 kann entnommen werden, dass ein zufälliges Pad $P \in \mathbb{G}_1$ mit zwei Komponenten (aufgrund der ElGamal Verschlüsselung) erstellt wird und dieses verwendet wird, um die verschlüsselte Stimme *c_bar* zu verblinden. So kann das Wahlamt über das Web Bulletin Board nicht mehr herausfinden, welche Stimme zu welcher Sitzung gehört hat, da es die Stimme nicht mehr erkennen würde. Das Pad selbst wird im Anschluss für die Signatur auch verblindet und die Parameter G_1, G_2, G_3 werden generiert.

Listing 4.1: Implementierung des Verblindens der Stimme und der dazugehörigen Parameter

```
1  def blind(self,vote,g):
2
3          P1=[]
4          P2=[]
5          c1_bar=[]
6          c2_bar=[]
7          P1_bar=[]
8          P2_bar=[]
9
10         for i in range(0,self.params):
11                 P1.append(self.pairing.random(G1))
12                 P2.append(self.pairing.random(G1))
13         P={'P1': P1, 'P2': P2}
14
15         for i in range(0,self.params):
```

```
16                  c1_bar.append((vote['c1'][i])* (P['P1'][i]**(-1)))
17                  c2_bar.append((vote['c2'][i])* (P['P2'][i]**(-1)))
18          c_bar={'c1_bar':c1_bar, 'c2_bar':c2_bar}
19
20          e=self.pairing.random(ZR)
21          f=self.pairing.random(ZR)
22          (f1,f2)=self.randdecomp(f)
23
24          for i in range(0,self.params):
25                  P1_bar.append(P['P1'][i]**e)
26                  P2_bar.append(P['P2'][i]**e)
27          P_bar={'P1_bar':P1_bar, 'P2_bar':P2_bar}
28
29          G={'G1':g**e, 'G2':g**(f1), 'G3':g**(e*f2)}
30          return (c_bar, P_bar, G, e, f1, f2)
```

Aufgrund des Verblindens ändern sich auch bei der Signatur einige Kleinigkeiten. In Listing 4.2 ist zu erkennen, dass der Signaturteil S in zwei Teile aufgeteilt wird. Es wird außerdem sowohl vom Client $(G1, G2, G3)$ als auch vom Aussteller des Zertifikats $(z1, z2)$ ein Rauschfaktor in die Signatur gestreut. So kann durch den Rauschfaktor seitens des Wahlamts später nicht mehr auf die zugehörige Signatur geschlossen werden.

Listing 4.2: Implementierung der AGHO Blindsignatur-Ausstellung

```
1 def sign(self,g, sk, c_bar,h, G, P_bar):
2
3          r=self.pairing.random(ZR)
4          (z1,z2)=self.randdecomp(sk['z'])
5          R_bar=g**r
6          S1_bar=(g**z1)*(G['G2']**(-(r*sk['v'])))*self.mprod(c_bar,
               sk['w'])
7          S2_bar=(G['G1']**z2)*(G['G3']**((-r)*sk['v']))*self.pprod(
               P_bar, sk['w'])
8          T_bar=h**(1/r)
9
10         sig={'R_bar':R_bar, 'S1_bar':S1_bar, 'S2_bar':S2_bar,'T_bar
               ': T_bar}
11
12         return (sig,z1, z2,r)
```

Diese Signatur wird anschließend vom Client zu einer klassischen AGHO Signatur $\sigma = (R, S, T)$ zusammengesetzt.

4.3.5 Die Zero-Knowledge Proofs

Im nächsten Schritt wurde ein Script für die ZKPs erstellt. Dieses Skript enthält die folgenden Funktionalitäten:

- Durchführen eines NIZKs über die korrekte Form der verblindeten Stimme
- Durchführen eines NIZKs über die korrekte Form der Signatur
- Durchführen eines NIZKs über die korrekte Entschlüsselung der Stimme und das Verwenden des richtigen Schlüssels der Auszählbehörde

- Verifikation der genannten Zero-Knowledge Proofs

Besonders interessant war hier das Implementieren des ZKPs für die Serverseite. Aufgrund eines Problems bei der Serialisierung von Werten aus $\mathbb{G}_2$ (genaueres in Abschnitt 4.3.6) wurde dieser ZKP etwas abgeändert implementiert. Wie in Listing 4.3 zu sehen ist, wurde vor dem Durchführen des Beweises für alle Gleichungen mit Elementen aus $\mathbb{G}_2$ das Pairing mit g nach $\mathbb{G}_T$ durchgeführt (siehe Listing 4.4). Der Zero-Knowledge Proof wurde also mit Elementen aus $\mathbb{G}_T$ durchgeführt, da in dieser Gruppe das Serialisierungsproblem nicht besteht. Diese Lösung dient nur als Übergangslösung für die prototypische Implementierung, um das bestehende Problem zu umgehen. Für den tatsächlichen Einsatz eines Wahlsystems sollte das Problem behoben werden.

Listing 4.3: Implementierung des Server-seitigen ZKPs

```
1  def ZKP_correctFormatS(h,g, sig, pk, G, c_bar, sk, z1, z2,ri, P_bar
       , params):
2          group=PairingGroup('BN254')
3          nu=[]
4          N=[]
5          r=[]
6          N_bar=""
7
8          for i in range(0,params*2+4):
9                  nu.append(group.random(ZR))
10
11         for i in range(0, params*2):
12                 N.append(mapToGT(g,h**nu[i]))
13         N.append(mapToGT(g,(h**nu[2*params])*(h**nu[2*params+1])))
14         N.append(g**nu[2*params+2])
15         N.append(mapToGT(g,sig['T_bar']**nu[2*params+2]))
16         tmp=(pk['V']**nu[2*params+2])*(h**(nu[2*params+3]))
17         N.append(mapToGT(g,tmp))
18         cprod=calcCprod(c_bar, nu, params)
19         N.append((g**nu[2*params])*(G['G2']**(nu[2*params+3]))*
               cprod)
20         pprod=calcPprod(P_bar,nu, params)
21         N.append((G['G1']**nu[2*params+1])*(G['G3']**(nu[2*params
               +3]))*pprod)
22
23         for i in range(len(N)):
24                 N_bar=N_bar+str(N[i])
25
26         ch=H(N_bar.encode('utf-8'))
27
28         for i in range(0, 2*params):
29                 r.append(ch*sk['w'][i]+nu[i])
30         r.append(ch*z1+nu[2*params])
31         r.append(ch*z2+nu[2*params+1])
32         r.append(ch*ri+nu[2*params+2])
33         r.append(ch*(-sk['v']*ri)+nu[2*params+3])
34
```

```
35          return (ch,r)
```

Listing 4.4: Implementierung der Mapping Funktion von Elementen aus $\mathbb{G}_2$ nach $\mathbb{G}_T$

```
1 def mapToGT(g,m):
2         group=PairingGroup('BN254')
3         N=group.pair_prod(g,m)
4 
5         return N
```

Auch für die Verifikation des Zero-Knowledge Proofs in Listing 4.5 wurde für die Gleichungen mit Elementen aus $\mathbb{G}_2$ wieder zuerst die Transformation nach $\mathbb{G}_T$ und anschließend die klassische Verifikation durchgeführt.

Listing 4.5: Implementierung der Server-seitigen ZKP Verifikaion

```
1 def verifyZKP_FormatS(h,g,pk, ch, r, c_bar, P_bar, G, sig, params):
2         v=[]
3         v_bar=""
4 
5         for i in range(0, 2*params):
6                 v.append(mapToGT(g,(h**r[i])*((pk['W'][i]**(-ch))))
                     )
7         v.append(mapToGT(g,(h**(r[2*params]))*(h**r[2*params+1])*((
             pk['Z']**(-ch)))))
8         v.append((g**r[2*params+2])*(sig['R_bar']**(-ch)))
9         v.append(mapToGT(g,(sig['T_bar']**r[2*params+2])*(h**(-ch))
             ))
10        v.append(mapToGT(g,(pk['V']**r[2*params+2])*(h**r[2*params
             +3])))
11        cprod=calcCprod(c_bar,r,params)
12        pprod=calcPprod(P_bar,r,params)
13        v.append((g**r[2*params])*(G['G2']**(r[2*params+3]))*cprod
             *(sig['S1_bar']**(-ch)))
14        v.append((G['G1']**r[2*params+1])*(G['G3']**(r[2*params+3])
             )*pprod*(sig['S2_bar']**(-ch)))
15 
16        for i in range(len(v)):
17                v_bar=v_bar+str(v[i])
18 
19        return ch==H(v_bar.encode('utf-8'))
```

4.3.6 Testen aller Komponenten

Um die erstellten Python Skripts auf ihre Funktionalität zu testen, wurden 4 Testskripts implementiert:

- *charmLibraryTest.py*:
  - Testen der Durchführbarkeit von Single Exponent Zero-Knowledge Proofs für Elemente aus $\mathbb{G}_1$, $\mathbb{G}_2$ und $\mathbb{G}_T$
  - Testen der Bilinearitätseigenschaft in beiden Argumenten

  - Testen der Skalarmultiplikationseigenschaft in beiden Argumenten
  - Testen der Exponentation in beiden Argumenten
  - Testen der Serialisierung und Deserialisierung in beiden Argumenten
- *ElGamaltests.py*
  - korrektes Ver- und Entschlüsseln mit nur der Stimme als Attribut
  - korrektes Ver- und Entschlüsseln mit zusätzlichen öffentlichen Attributen
- *AGHOTests.py*
  - korrektes Signieren und Funktionalität der Verifikation der AGHO Signatur mit nur der Stimme als Attribut
  - korrektes Signieren und Funktionalität der Verifikation der AGHO Signatur mit zusätzlichen öffentlichen Attributen
  - korrektes Signieren und Funktionalität der Verifikation der AGHO Blindsignatur mit nur der Stimme als Attribut
  - korrektes Signieren und Funktionalität der Verifikation der AGHO Blindsignatur mit zusätzlichen öffentlichen Attributen
- *ZKPTests.py*
  - Durchführen und Verifizieren des ZKPs für den Client über die korrekte Form der Stimme mit nur der Stimme als Attribut
  - Durchführen und Verifizieren des ZKPs für den Client über die korrekte Form der Stimme mit zusätzlichen öffentlichen Attributen
  - Durchführen und Verifizieren des ZKPs für das Wahlamt über die korrekte Form der Signatur mit nur der Stimme als Attribut
  - Durchführen und Verifizieren des ZKPs für das Wahlamt über die korrekte Form der Signatur mit zusätzlichen öffentlichen Attributen
  - Durchführen und Verifizieren des ZKPs für die Auszählbehörde über die korrekte Form der Stimme mit nur der Stimme als Attribut
  - Durchführen und Verifizieren des ZKPs für die Auszählbehörde über die korrekte Form der Stimme mit zusätzlichen öffentlichen Attributen
  - Der Aufruf der oben genannten ZKPs über die jeweiligen Bibliotheken (ElGamal bzw. AGHO Blindsignatur) und deren Funktionalität

Wie im vorhergehenden Abschnitt beschrieben, gab es bei der Durchführung des Single Exponent ZKPs Probleme bei Elementen aus der Gruppe $\mathbb{G}_2$. In Listing 4.6 ist der Test für diese Gruppe abgebildet. Es wurde der klassische Schnorr Single Exponent Proof durchgeführt.

Listing 4.6: Implementierung der Single Exponent ZKP Tests für Elemente aus $\mathbb{G}_2$

```
1  def testZKPG2(self):
2          groupObj=PairingGroup('BN254')
3          params=1
4          el = ElGamal(params)
5          agho = AGHOBlind(el)
6          h=groupObj.random(G2)
7          (sk, pk)=agho.keygen(h)
8          (ch,r)=self.ZKP_exponentG2(h,sk['v'], pk['V'])
9          isTrue=self.ZKP_exponentG2_verify(ch,r, pk['V'],h)
10         print("Single exponent ZKP proof in G2", isTrue)
```

```
11
12 def ZKP_exponentG2(self,h, sk, pk):
13         group=PairingGroup('BN254')
14         nu=group.random(ZR)
15         N=h**nu
16         ch=group.hash(group.serialize(N),ZR)
17         r=ch*sk+nu
18         return (ch,r)
19
20 def ZKP_exponentG2_verify(self,ch,r,pk,h):
21         group=PairingGroup('BN254')
22         N=(h**r)*(pk**(-ch))
23         return ch==group.hash(group.serialize(N),ZR)
```

In Abbildung 4.3 sind alle Ergebnisse der Tests aus *charmLibraryTest.py* zu sehen. Es kann eindeutig entnommen werden, dass der Test mit dem Single Exponent Proof nur für Elemente aus der Gruppe $\mathbb{G}_2$ fehlschlägt.

```
Andreas-MacBook-Pro:testRest andi$ python3 charmLibraryTests.py
Exponentiation equation (g**(k1+k2)==(g**k1)*(g**k2)) in G1 True
.Exponentiation equation (g**(k1*k2)==(g**k1)**k2) in G1 True
.Exponentiation equation (h**(k1+k2)==(h**k1)*(h**k2)) in G2 True
.Exponentiation equation (h**(k1*k2)==(h**k1)**k2)  in G2 True
.Linearity in 1st Argument: True
.Linearity in 2nd Argument: True
.Scalar multiplication in 1st Argument: True
.Scalar multiplication in 2nd Argument: True
.Scalar multiplication in both Arguments: True
.Serialization Test Result G1 True
.Serialization Test Result G2 True
.Single exponent ZKP proof in G1 True
.Single exponent ZKP proof in G2 False
.Single exponent ZKP proof in GT True
```

Abbildung 4.3: Testergebnis des Single Exponenten ZKPs über Elemente aus $\mathbb{G}_2$

4.4 Prototypische Implementierung eines Client-Server Modells

4.4.1 Infrastruktur und Aufbau des Wahlsystems

Wie schon in Abschnitt 4.1 erwähnt sind für dieses Protokoll drei Entitäten von Nöten: Der Client C, das Wahlamt W und die Auszählbehörde A. Bei der Implementierung werden für die Einfachheit das Wahlamt und die Auszählbehörde zu einer Ressource zusammengefasst. Diese zwei Entitäten werden aber trotzdem über verschiedene Protokollendpunkte angesprochen werden und verwenden unterschiedliche Ressourcen. Folgendermaßen ist das prototypische Wahlsystem aufgebaut (siehe auch Abbildung 4.4):

- Als Programmiersprache sowohl für den Client als auch für den Server wurde Python3 gewählt.
- Der Server stellt sowohl das Wahlamt als auch die Auszählbehörde dar und läuft in einer virtualisierten Umgebung.
- Das Frontend der Applikation ist der Client, welcher über ein Kommandozeilen-Interface bedienbar ist.

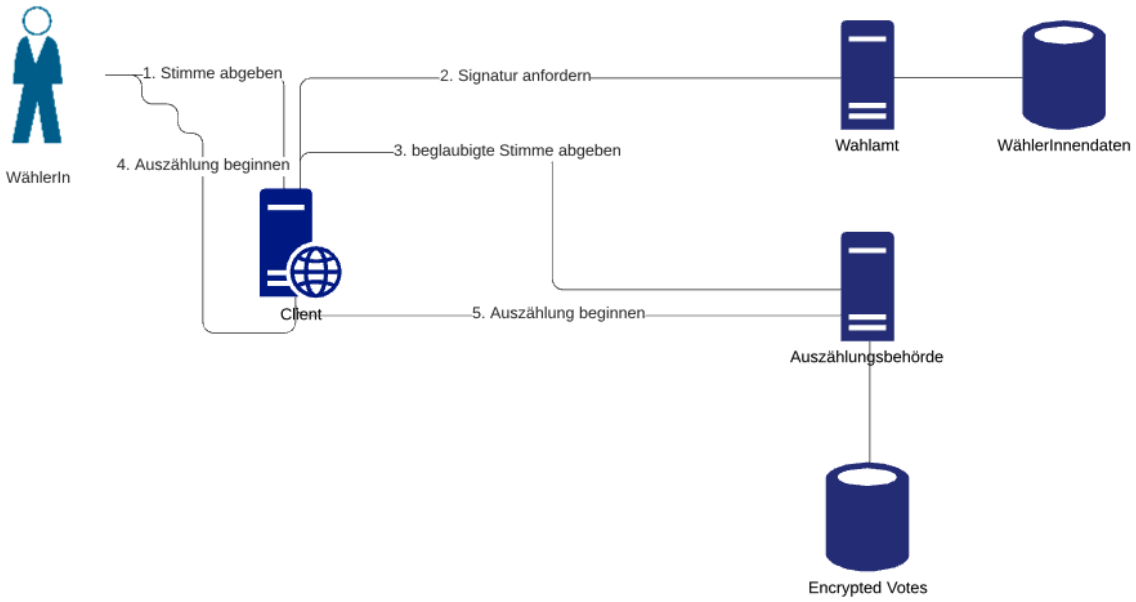

Abbildung 4.4: Aufbau des Wahlsystems (erstellt mit LucidChart [Luc])

- Die Kommunikation der beiden Komponenten findet über eine REST-Schnittstelle statt.
- Für die Datenbanken wurden SQLite Datenbanken gewählt.

4.4.2 Voraussetzung und Installation

Um die im Zuge dieser Masterarbeit entstandene prototypische Implementierung des Client-Server Modells verwenden zu können, müssen folgende Voraussetzungen erfüllt sein:

- Installation von Python3
- Installation der Charm Bibliothek (Funktionsweise siehe [AMG+11])
- Installation von virtualenv (Virtuelle Umgebung für Python)

```
1 pip3 install virtualenv
```

- Installation von flask (RESTFul Programmieren mit Python) [Aga17]

```
1 pip3 install flask flask-jsonpify flask-sqlalchemy flask-
    restful
```

- Installation der Requests-Bibliothek für das Durchführen von Client-seitigen RESTFul Serveranfragen:

```
1 pip3 install requests
```

- Installieren eines DB Browsers für SQLite (Empfohlen für das Erstellen von Datenbanken)
- Insgesamt werden 4 SQLite Datenbanken im selben Ordner wie der Server mit folgenden Namen benötigt
  - AKeyData (zur Speicherung der ElGamal Schlüssel)

  - WKeyData (zur Speicherung der Signaturschlüssel)
  - WVoterData (die Datenbank, welche die Wählerdaten enthält)
  - AVoteData (die Datenbank in der die EVs samt Signaturen abgelegt werden)

Um die Applikation zu starten, muss für den Server eine virtuelle Umgebung geschaffen werden, also für einen ausgewählten Ordner *bspdir* folgende Befehle durchgeführt werden

```
1 virtualenv bspdir
2 source bspdir/bin/activate
```

Anschließend müssen die Bibliotheken (Charm, flask, und die im Zuge dieser Masterarbeit entstandene AGHO Bibliothek) auch hier noch einmal hineingezogen werden, damit sie auch in der virtuellen Umgebung vorhanden sind.
Der Server (also das Wahlamt bzw. die Auszählbehörde) kann dann folgendermaßen gestartet werden.

```
1 python3 BackendVoteAPI.py
```

Der Server hört nun auf *localhost:5002*. es werden beim erstmaligen starten automatisch zwei Testbenutzer mit den folgenden Zugangsdaten erzeugt:

- User 1:
  - **Username:** testuser1
  - **Passwort:** supersecurepassword123
- User 2:
  - **Username:** testuser2
  - **Passwort:** reallysecurepassword123

Der Client kann folgendermaßen gestartet werden:

```
1 python3 PythonVoterAPI.py
```

4.4.3 Mitgelieferte Dateien

Folgende Dateien sind mit dieser Masterarbeit für die Bedienbarkeit des Client-Server Modells mitgeliefert:

- *kryptlib/*
  - *ElGamalImpl.py*: Die Implementierung der ElGamal Ver- und Entschlüsselung
  - *AGHOSignature.py*: Die Implementierung der AGHO Signatur
  - *AGHOBlind.py*: Die Implementierung der AGHO Blindsignatur
  - *ZKP.py*: Die Implementierung der Zero-Knowledge Proofs
- *charmLibraryTests.py*: Tests für die Funktionalität der Charm Bibliothek
- *ElGamaltests.py*: Tests für die Funktionalität der ElGamal Ver- und Entschlüsselung
- *AGHOTests.py*: Tests für die AGHO Signatur und die AGHO Blindsignatur
- *ZKPTests.py*: Tests für die Funktionalität der ZKPs
- *PythonVoterAPI.py* Der Client der prototypischen Implementierung
- *BackendVoteAPI.py* Der Server der prototypischen Implementierung

4.4.4 Technische Daten des Testgerätes

Im folgenden werden die technischen Daten des Geräts und die Versionen der verwendeten Bibliotheken angeführt mit dem die Funktionalität aller Applikationen und Tests überprüft wurde.

MacBook Pro (15-inch, 2018)
macOS Mojave Version 10.14.6
Prozessor: 2,6 GHz Intel Core i7
Speicher: 16GB 2400 MHz DDr4
Grafikkarte: Radeon Pro 560X 4GB, Intel UHD Graphics 630 1536 MB

Python Version: 3.7.5
Flask Version: 1.1.2
SQLite Version: 3.24.0 2018-06-04 14:10:15 -
95fbac39baaab1c3a84fdfc82ccb7f42398b2e92f18a2a57bce1d4a713cbaapl

4.4.5 Getroffene Annahmen für die Implementierung

Im Folgenden wird beschrieben welche Annahmen für die prototypische Implementierung dieses Client-Server Modells getroffen wurden. Diese Auflistung soll das Bewusstsein über diese Einschränkungen aufzeigen und darauf hinweisen, dass bei einer Weiterführung dieser Implementierung auf die folgenden Punkte geachtet werden muss.

- Es wird davon ausgegangen, dass die WählerInnen ihre Zugangsdaten bereits (zum Beispiel per Post) erhalten haben.
- Das Wahlamt und die Auszählungsbehörde werden auf einen Server zusammengefasst.
- Es muss davon ausgegangen werden, dass sowohl das Wahlamt als auch die Auszählbehörde ehrlich sind. Das bedeutet, dass von Seiten des Wahlamts als auch der Auszählbehörde kein *Denial Of Service* durchführt wird und dass das Wahlamt keine doppelten Wahlen (doppelte Signaturausstellung) trotz Korrektheit der Datenbankeinträge zulässt.
- Die Auszählung der Wahl kann durch einen Menüpunkt in der prototypischen Implementierung direkt vom Wähler / von der Wählerin vorgenommen werden. Dies ist nur für Testzwecke gedacht und sollte später durch einen anderen Mechanismus (z.B. Administratorrollen) ersetzt werden.
- Auch die Möglichkeit die Wahl zurückzusetzen ist dafür da, um das Testen zu vereinfachen und ist nicht für den tatsächlichen Einsatz gedacht.
- Es ist nicht möglich mehrere Wahlen gleichzeitig laufen zu lassen.
- In dieser konkreten Implementierung wird kein Schwellwert-Kryptosystem verwendet.
- Es wird davon ausgegangen, dass sich die WählerInnendaten bereits in der Datenbank befinden und dass diese Einträge korrekt sind. Es werden automatisch zwei Testuser erzeugt.
- Es wird keine Replikation der Server und der Datenbanken geben.
- Der Server ist so implementiert, dass er pro Wählerin nur eine Stimme signiert. Das bedeutet, dass nur Wahlen durchgeführt werden können, welche genau eine Auswahl benötigen. Mehrere Stimmen sind nicht zulässig.

- Es wird davon ausgegangen, dass die User ehrlich bezüglich Angaben von öffentlichen Attributen (Alter, Geschlecht,...) sind, da diese Angaben von den WählerInnen angegeben werden. Es wird lediglich darauf überprüft, ob die Angaben gültig sind. Das bedeutet, es wird zum Beispiel überprüft, ob das Geschlecht entweder männlich oder weiblich ist.

4.4.6 Die Implementierung

In diesem Abschnitt wird die prototypische Client-Server Implementierung in den einzelnen Schritten beschrieben und der Aufbau dieser vorgestellt.

Allgemeiner Aufbau

Die Kommunikation zwischen dem Client und dem Server findet über eine REST Schnittstelle statt. Der Benutzer oder die Benutzerin selbst hat folgende Auswahlmöglichkeiten über den Client gegeben (siehe auch Abbildung 4.5):

1. Eine Stimme abgeben
2. Auszählung durchführen (nur für Testzwecke)
3. Web Bulletin Board anzeigen lassen
4. Wahl-Status zurücksetzen (nur für Testzwecke)
5. Client verlassen

Das Wahlamt bietet folgende Schnittstellen für den Client an:

- */pkEV*: GET Methode für das Holen des öffentlichen Schlüssels für das Verschlüsseln der Stimme
- */sign*: POST Methode für das Signieren einer verblindeten Encrypted Vote
- */reset*: GET Methode für das Zurücksetzen des Wahlstatus

Die Auszählbehörde bietet die folgenden Schnittstellen:

- */pkSig*: GET Methode für das Holen des öffentlichen Signaturschlüssels
- */vote*: POST Methode für das Abgeben einer EV inklusive AGHO Signatur
- */count*: GET Methode für Auszählen der EVs
- */wbb*: GET Methode für das Erhalten der Wahldaten für die Darstellung am WBB
- */reset*: GET Methode für das Zurücksetzen des Wahlstatus

Die Schnittstellen für das Zurücksetzen der Wahl sind in der Implementierung für beide Serverentitäten zusammengefasst worden.

```
Andreas-MacBook-Pro:testRest andi$ python3 PythonVoterAPI.py
------What would you like to do?------
[1] Vote
[2] Counting
[3] Web Bulletin Board
[4] Reset Vote-status
[5] Quit Client
Notice that this is a test-application for the protocol
->
```

Abbildung 4.5: Hauptmenü des Clients

Abgeben einer Stimme

Die große Anzahl an Zwischenschritten (Verschlüsseln, Verblinden, Signieren, Zero-Knowledge Proofs erstellen und verifizieren), die beim Abgeben einer Stimme durchgeführt werden müssen, sind in dieser Implementierung komplett vor den WählerInnen versteckt. Wie in Abbildung 4.6 zu sehen ist, muss ein Benutzer oder eine Benutzerin einfach das öffentliche Attribut *Geschlecht* angeben und die Auswahl treffen. Außerdem muss er/sie seine/ihre Zugangsda-

```
----Please enter your choice---
[1]  male
[2]  female
[3] quit
->
1
----Please enter your choice---
[1]  Eric Example
[2]  Max Mustermann
[3]  Conrand Candidate
[4] quit
->
3
Please enter your username
testuser
please enter your password
testpwd
Vote submitted successfully
------What would you like to do?------
[1] Vote
[2] Counting
[3] Web Bulletin Board
[4] Reset Vote-status
[5] Quit Client
Notice that this is a test-application for the protocol
```

Abbildung 4.6: Ablauf der Stimmabgabe aus User-Sicht

ten, die für die Authentifizierung gegenüber dem Server gebraucht werden, angeben. Hat der User / die Userin all diese Daten übergeben, so werden folgende Schritte im Hintergrund durchgeführt:

- Verschlüsseln und Verblinden der Stimme und Erstellen eines NIZKs über die Korrektheit der Stimme
- Senden des Stimmzettels an das Wahlamt
- Das Wahlamt führt folgende Schritte durch
  - Überprüfen der Gültigkeit der Zugangsdaten
  - Überprüfen der Korrektheit des ZKPs
  - Überprüfen, ob der Wähler / die Wählerin wählen darf (ob er/sie wahlberechtigt ist und ob er/sie bereits eine Stimme signieren lassen hat)
  - Aktualisierung des Status in der WählerInnen-Datenbank

  Sollten alle der oben genannten Komponenten positiv ausfallen, so signiert das Wahlamt die Stimme und schickt die Signatur zusammen mit dem ZKP über die Korrektheit der Signatur an den Client zurück (andernfalls wird eine Fehlermeldung zurückgegeben).
- Der ZKP wird anschließend verifiziert und die Stimme wird entblindet und samt der entblindeten Signatur an die Auszählbehörde geschickt.

- Die Auszählbehörde
  - überprüft die Verifikationsgleichungen für die Signatur
  - überprüft anhand der Signatur, ob diese Encrypted Vote schon einmal vorgelegt wurde, um doppelte Stimmabgaben zu vermeiden
  - legt die Encrypted Vote (inklusive Signatur) bei Gültigkeit in der Stimmen-Datenbank ab

Sollte eine Person bereits gewählt haben, oder die Auszählung der Wahl bereits stattgefunden haben, so schickt der Server einen Fehlercode zurück und die Stimmabgabe wird abgelehnt (siehe Abbildung 4.7)

```
----Please enter your choice---
[1]  male
[2]  female
[3] quit
->
1
----Please enter your choice---
[1]  Eric Example
[2]  Max Mustermann
[3]  Conrand Candidate
[4] quit
->
3
Please enter your username
testuser
please enter your password
testpwd
An error happened during signature 401 You are not entitled to vote or have already voted
```

Abbildung 4.7: Mechanismus zur Vermeidung von doppeltem Wählen

Auszählung durchführen

Um die Auszählung durchzuführen sendet der Client eine passende Anfrage an die Auszählbehörde. Diese holt sich alle Stimmen aus der Datenbank und entschlüsselt sie. Die Stimmen werden dann mit einer Lookup-Tabelle verglichen und so ausgezählt. Der Client bekommt die Anzahl der erhaltenen Stimmen pro Kandidaten von der Auszählbehörde zurück und errechnet sich daraus dann das Ergebnis. Dieses kann wie in Abbildung 4.8 abgelesen werden.

```
------What would you like to do?------
[1] Vote
[2] Counting
[3] Web Bulletin Board
[4] Reset Vote-status
[5] Quit Client
Notice that this is a test-application for the protocol
->
2
-------Voting-Results------
Eric Example :  0
Max Mustermann :  0
Conrand Candidate :  2
The winner is:  Conrand Candidate
```

Abbildung 4.8: Auszählung der Wahl aus User-Sicht

Web Bulletin Board anzeigen

Das Web Bulletin Board soll dazu dienen die Stimmabgaben nachvollziehen zu können. Deswegen werden dort alle Encrypted Votes, die entschlüsselten Stimmen, die Signaturen, sowie die Zero-Knowledge Proofs und auch die Verifikationen dieser angezeigt (siehe Abbildung 4.9). Um keine Teilergebnisse vorab bekannt zu geben, ist das Abrufen des Boards erst nach der Auszählung möglich.
Die Daten, welche hier angezeigt werden, werden von der Auszählbehörde folgendermaßen zur Verfügung gestellt:

- Encrypted Vote aus der Datenbank holen
- Entschlüsseln der Stimmen,
- Erstellen der ZKPs über die einzelnen Stimmen
- senden dieser Daten an den Client

Der Client führt anschließend die Verifikation über die ZKPs durch. Die verschlüsselten Stim-

```
------What would you like to do?------
[1] Vote
[2] Counting
[3] Web Bulletin Board
[4] Reset Vote-status
[5] Quit Client
Notice that this is a test-application for the protocol
->
3
------WEB-BULLETIN-BOARD------
------------------------------
-------Encrypted-Votes--------
[['1:IB1Zp5uJwHTUq5KCNFnD2reVnVCSOdyTNTsLyAkZR2UB', '1:C9PCzz7XZN2XQzINcNWraerFR9LfyOquuzK8QD9CPkgB',
 '1:FkHr7+mUsLTtPldBrJwHmitLDzkboZnd6IYVbAl6RCgB', '1:HxTztwKSYw+k1BCsklya50/+b8pOzRmcRpyDlVECq8IA'],
 ['1:DByF/Cjf1etNkSQsQfFcs8wnrV/Mq5mZDK8q1mC+ypIA', '1:GUTYiZ7FxCyGry0+gisp0bUsR3VTxLLHwfe8c8bEP7YB',
 '1:DB5lV7VIaAQ/YiXRIvanCO8WHg5/tnGO+byaYPmtu+EA', '1:HOBnJ6ZtoM/M3xqeFOtFgBfEcbgZxEIQXlNye53iu2IB']]
----------Signatures----------
[['1:DlMwXsBk84d8AtetQErU+uDMQIgYiRrFxo+AGxs6QD0A', '1:Il6ESgx3e6JfyUBvzBesP+D5JcVqgvKKmo3d5Wr4ZxAB',
 '2:IpVMmkpeA/LHLaTCWYd0QxMC4LnvS/yuTnU5vUbl8UUG2gvFHHz4AzVmd58f8mShf1HoEIbl86jzJhplZ2jkEQE='], ['1:G
GKlUZRVSbNMeR4Cr/ZJyPd/0D/C40fWSzNde8kOqk4A', '1:CZC4MfJPZDUBPrGTgBiUOH7hSGoLnvyRi4ibXmmMGJ0B', '2:Ew
0f7D2kxjjqdoD/n7sKnh7Mn7x1N0NaQ+qFzOdWNsIb/s2SjtTAbzbYL8YZq7Iq4xuKcgiElyYWc59XiHBFnwE=']]
--------Decrypted-Votes-------
[['male', 'Eric Example'], ['female', 'Max Mustermann']]
-----Zero-Knowledge-Proof-----
['0:FMN1hy9i9PaG27zPLe9ioykp+gNu20ex5Vtji5UyNIA=', '0:GgmjwgZI3GBlOJX+33zhlwP7RX3W5n/qSuJk448N4KA=']
['0:CqM7fK7IVKmDy3SPHDjbaTP2u9FkXTqSXi6TL6GTUXM=', '0:GBGfGGo/gSwFWhpQ3c5GuQA7llQrFYa7Ca+mogwhdbg=']
-------Verifications----------
[True, True]
```

Abbildung 4.9: Das Web Bulletin Board aus User-Sicht

men und die ZKP-Komponenten werden am WBB im base64 Format angegeben, da dieses Format übersichtlicher ist, als jenes in der Pairing-Darstellung.

Wahlstatus zurücksetzen

Um den Wahlstatus zurückzusetzen wird vom Client eine GET-Anfrage an den Server geschickt. In diesem Fall werden die Funktionalitäten der Anfrage für das Wahlamt und die Auszählbehörde zusammengefasst, da dieser Menüpunkt nur für Testzwecke eingebaut wurde. Für das Zurücksetzen der Wahl werden folgende Schritte durchgeführt:

- Leeren der Stimmen-Datenbank, welche alle Encrypted Votes beinhaltet
- Zurücksetzen aller Status der Wähler auf „noch nicht gewählt“
- Status der Wahl auf nicht beendet setzen

4.4.7 Die IT-Sicherheitsbedingungen bezogen auf den implementierten Prototypen

In diesem Abschnitt wird die in Abschnitt 4.2 durchgeführte Überprüfung der Bedingungen erneut für die prototypisch implementierte Variante durchgeführt. Die Ergebnisse werden anschließend in Tabelle 4.1 zusammengefasst.

Verfügbarkeit

In dieser Arbeit wurde eine virtuelle Umgebung verwendet und keine Form der Replikation eingebaut. Sollte hier der Server ausfallen, so gibt es kein Backup. Eine Redundanz in das System zu bringen, ist aber durchaus möglich. Die Bedingung ist daher nicht erfüllt, ist aber durch eine Erweiterung des Systems erfüllbar.

Anonymität

Wie bereits in Abschnitt 4.2 beschrieben wird aufgrund der Blindsignatur die Verbindung zwischen den Encrypted Votes und dem, was das Wahlamt, welches die Signatur ausstellt, zu sehen bekommt, unterbunden. Das Wahlamt kennt zwar die verblindete Stimme und kann sie aufgrund der Userdaten einer Person zuordnen, erkennt aber später die EV am Web Bulletin Board nicht mehr.
Auch die Auszählbehörde wurde so implementiert, dass bei der Stimmabgabe keine Authentifizierung benötigt wird.
Somit ist die Bedingung dann komplett erfüllt, wenn bei der Stimmübertragung die Anonymität auch gewährleistet wird. Ein Beispiel dafür wäre der Tor Browser.

Wähler Nachweisbarkeit

Da am Web Bulletin Board später sowohl die Encrypted Votes samt Signaturen als auch die unverschlüsselten Stimmen veröffentlicht werden, wäre es für eine Wählerin möglich ihre eigene Stimme wieder zu finden. Um die Stimmzettel eindeutig identifizieren zu können, wird die dazugehörige Signatur gebraucht, welche auch auf dem Web Bulletin Board zu finden ist. Die Bedingung ist somit erfüllt.

Universelle Nachweisbarkeit

Dadurch, dass die Stimmen und die Encrypted Votes am Web Bulletin Board veröffentlicht werden, kann jede Person überprüfen, ob die Auszählung korrekt durchgeführt wurde. Somit ist die Bedingung der universellen Nachweisbarkeit erfüllt.

Integrität

Wie bereits in Abschnitt 4.2 beschrieben ist diese Bedingung durch den Aufbau des Protokolls erfüllt.

Wählbarkeit

In der prototypischen Implementierung wurde in der WählerInnen-Datenbank vermerkt, ob ein Wähler bzw. eine Wählerin wahlberechtigt ist. Außerdem befindet sich in dieser Daten-

bank auch ein Eintrag, der angibt, ob eine Person bereits eine Stimme abgegeben hat, oder nicht. Durch diese beiden Werte ist es einem Wähler oder einer Wählerin nicht möglich zwei Stimmen vom Wahlamt W signieren zu lassen und dadurch zwei Stimmen an die Auszählbehörde zu übergeben.
Auch die Auszählbehörde überprüft vor dem Annehmen eines Stimmzettels, ob dieser (genauer gesagt, die dazugehörige Signatur) schon einmal abgegeben wurde. Dadurch wird verhindert, dass eine Person eine signierte Stimme öfter an die Auszählbehörde schicken kann. Diese Bedingung ist also erfüllt.

Keine Belegbarkeit

Da dieses Protokoll sehr großen Wert darauf legt die einzelnen Attribute für statistische Auswertungen verwenden zu können, und da die Stimmen am Web Bulletin Board veröffentlicht werden, um die Universelle Nachweisbarkeit zu erfüllen, ist diese Bedingung nicht erfüllt. Am Ende der Wahl werden sowohl die Encrypted Votes als auch die unverschlüsselten Stimmen und die Signaturen am öffentlich zugänglichen Web Bulletin Board veröffentlicht. Somit können die WählerInnen ganz einfach beweisen, was sie gewählt haben.

Robustheit

Durch die Form der restriktiven AGHO Blindsignatur ist diese Bedingung erfüllt. Es wird die Stimme nur bei Gültigkeit signiert. Das bedeutet, dass die Encrypted Vote die richtige Form haben muss, damit ein Zertifikat auf diese Stimme ausgestellt wird. Durch den Aufbau des Clients werden nur die Stimmen, welche zur Auswahl stehen, zugelassen. Durch den Wahlserver ist es nicht möglich öfter als einmal zu wählen. Außerdem wird bei einer Stimme, welche von einem unehrlichen Client abgegeben wird und keine der zur Auswahl stehenden KandidatInnen beinhaltet, nicht zur Wertung gezählt. Somit ist dieses System robust gegen falsche oder fehlerhafte Stimmen.

Fairness

Da die ver- und entschlüsselten Stimmen erst nach der Wahl (bei der Auszählung) am Web Bulletin Board veröffentlicht werden, können von den bereits abgegebenen Stimmen keine Teilergebnisse von anderen Personen errechnet werden.
Der Wahlserver signiert keine Stimmen mehr sobald die Auszählung begonnen hat und die Wahl damit als abgeschlossen gilt. Somit ist es nicht möglich nach der Auszählung noch eine Stimme abzugeben. Da dieser Wert aber nur in einer Variablen und nicht in einer Datenbank abgelegt wird, ist diese Bedingung nur dann erfüllt, wenn der Server während einer Wahl nicht neu gestartet wird. Um auch die Komponente der Menschen miteinzubeziehen sollte ein Schwellwert Kryptosystem verwendet werden. So besteht nicht mehr der Verlass auf eine einzelne Komponente.

Korrektheit

Wie in Abschnitt 4.2 beschrieben ist diese Bedingung erfüllt.

Zwang zur Wahl

Diese Bedingung ist bei Remote Systemen generell nur sehr schwer erfüllbar. Da ein Erpresser durch den Aufbau der Wahl eine Stimme am Web Bulletin Board wieder erkennen könnte, wenn diese nur einmal vorkommt, ist diese Bedingung nicht erfüllt.

Tabelle 4.1: Erfüllbarkeit der Bedingungen aus Abschnitt 2.1 für das implementierte Protokoll

Bedingung	Erfüllbarkeit
Verfügbarkeit	durch Erweiterung erfüllbar
Anonymität	unter Verwendung eines Onion Routers erfüllt
Wähler Nachweisbarkeit	erfüllt
Universelle Nachweisbarkeit	erfüllt
Integrität	erfüllt
Wählbarkeit	erfüllt
Keine Belegbarkeit	nicht erfüllt
Robustheit	erfüllt
Fairness	für ehrliche Teilnehmer erfüllt; durch Verwendung eines Schwellwert-Kryptosystems auch für eine beschränkte Anzahl an unehrlichen Teilnehmern erfüllbar
Korrektheit	erfüllt
Zwang zur Wahl	nicht erfüllbar ohne starke Änderungen

4.4.8 Erweiterungsmöglichkeiten

Da es sich bei der Implementierung, welche in Abschnitt 4.4.6 beschrieben wird, um eine prototypische Implementierung handelt, wird im Folgenden darauf eingegangen, welche Erweiterungsmöglichkeiten durch diese Masterarbeit entstehen.

Sicherheits-relevante Aspekte

Aus IT-Sicherheits-Sicht können noch einige Aspekte ergänzt werden. Es wird bei dieser Implementierung mit einem einzigen Schlüssel für die Encrypted Votes gearbeitet. Damit die Korrektheit und auch das Bestehen des Wahlgeheimnisses nicht von einer einzigen Person abhängt, kann dieses System auf ein Schwellwert-Kryptosystem erweitert werden.
Auch die Speicherung der ElGamal Schlüssel und der Signaturschlüssel sollte nicht im Klartext in einer einfach zugänglichen Datenbank erfolgen.
Die Passwörter der WählerInnen können durch deren Hashes ersetzt werden, um anschießend nur noch die Hashes zu vergleichen und zu umgehen, dass der Server sich das Passwort der UserInnen merkt.
Es sollte außerdem für den tatsächlichen Einsatz des Systems eine Inputvalidierung für Parameter durchgeführt werden, welche für Datenbankabfragen verwendet werden (Benutzername und Passwort).
Der Wahlstatus wird in einer globalen Variable hinterlegt. Sollte der Server während einer Wahl abstürzen, so ist es möglich, dass der Status zurückgesetzt wird und dadurch nach der Auszählung der Wahl weitere Stimmen abgegeben werden können. Durch Speicherung dieses Wertes in einer Datenbank kann das verhindert werden.
In dieser Implementierung wurde es dem User überlassen die öffentlichen Attribute anzugeben. Somit ist es zum Beispiel einer weiblichen Wählerin möglich anzugeben, dass sie männ-

lich ist. Um dies zu vermeiden könnten diese Attribute automatisch in die Stimme eincodiert werden. Dafür müsste zusätzlich die WählerInnen Datenbank um diese Attribute erweitert werden.
Da es sich bei der Implementierung um eine prototypische Implementierung handelt wurde keine Funktionalität eingebaut, welche es erlaubt ein Schlüsselpaar zu erneuern bzw. neu zu generieren. Die Serverkomponenten könnten um diese Funktionalität erweitert werden.
Zusätzlich zum Benutzernamen und dem Passwort sollte eine zweite Komponente, wie zum Beispiel ein *One Time Password*, für die Authentifizierung gegenüber dem Server hinzugezogen werden. Das kann zum Beispiel mit Hilfe des Microsoft Authenticators umgesetzt werden [Aut].
Außerdem sollten die Auszählbehörde und der Wahlserver auf zwei verschiedene Ressourcen aufgeteilt werden, da diese Ressourcen getrennt behandelt werden sollten. Weiters sollte bei einer tatsächlichen Webanwendung eine Transportverschlüsselung für die Datenübertragung verwendet werden, um die Vertraulichkeit zu gewährleisten.
Sollte es zwischen Signieren der Stimme und Abgeben bei der Auszählbehörde zu einem Fehler kommen, so sollte es die Möglichkeit geben (zumindest, wenn der Fehler serverseitig - Wahlbehörde - passiert), dass der Wählerstatus in der Datenbank zurückgesetzt wird. Das bedeutet, wenn die Wählerin keine Signatur bekommt, dann sollte in der Datenbank auch nicht vermerkt sein, dass die Wählerin bereits gewählt hat.

Erweiterung der Benutzerfreundlichkeit

Für die prototypische Implementierung des Clients wurde ein einfaches Kommandozeilen Interface für die Interaktion mit den BenutzerInnen gewählt. Der Client könnte auf eine Browser-basierte Anwendung erweitert werden. Ein Beispiel für so eine Anwendung wäre Angular mit Typescript und Node.js. Die im Zuge dieser Masterarbeit erstellte Bibliothek für die AGHO Blindsignatur könnte trotzdem weiterverwendet werden, da es von Node.js die Möglichkeit gibt mit Python zu kommunizieren. Es wird dabei mit Hilfe von Node.js über eine Schnittstelle auf Python-Seite zwischen den zwei Komponenten kommuniziert. Dafür kann die *child_process* Standard Bibliothek verwendet werden mit der einfach ein Python Script abgerufen werden kann (siehe [Kam15]).
Im Zusammenhang damit kann ein ansprechendes grafisches User Interface mit übersichtlichem Web Bulletin Board und vergangenen Ergebnissen erstellt werden.
Auch für einen Webservice, wie zum Beispiel Spring Boot mit Java, gibt es die Möglichkeit Python Skripts in Java auszuführen. Dies kann durch eine Bibliothek geschehen (siehe [oMH13]).
Für die Datenbanken könnten Schnittstellen hinzugefügt werden, die es erlauben Userdaten zu laden. Es wäre außerdem von Vorteil, wenn der Service um die Funktionalität erweitert werden würde, dass die entschlüsselten Stimmen gespeichert werden. Dadurch müssen sie für das Web Bulletin Board nur aus der Datenbank geholt werden, und müssen nicht jedes Mal neu berechnet werden, was bei einer großen Anzahl an WählerInnen einen Performancevorteil mit sich bringt.
Eine letzte interessante Erweiterungsmöglichkeit wäre, dass es den WählerInnen bei bestimmten Wahlen ermöglicht werden sollte, mehrere Stimmen abgegeben zu können. Dies wäre leicht umsetzbar, wenn zum Beispiel die Anzahl der bereits abgegebenen Stimmen in der Wählerdatenbank vermerkt werden würde. Dadurch könnte eine Multiple Choice Wahl abgehalten werden.

5 Ergebnisse und Schlussfolgerung

In dieser Masterarbeit wurden einige existierende Protokolle verglichen und ein neues Protokoll vorgestellt und implementiert. Der Vergleich der existierenden Protokolle zeigte, dass es bereits viele Ansätze und verschiedene Formen für elektronische Wahlsysteme gibt. Jedes der vorgestellten Wahlsysteme hat seine eigenen Vorteile und Nachteile. Ob es geeignet für eine elektronische Wahl ist hängt stark von dem Anwendungsgebiet ab. Gewisse Bedingungen sind für eine landesweite Wahl von enormer Wichtigkeit, spielen bei einer firmeninternen Wahl aber womöglich keine Rolle.
Der Vergleich der vorgestellten Systeme zeigte, dass es noch kein Protokoll gibt, welches alle Bedingungen erfüllt. Vor allem Remote Systeme haben die Schwachstelle, dass sie den Zwang zur Wahl, also das Vermeiden, jemanden zu einer Auswahl zwingen zu können, nicht erfüllen. Es konnten aber einige Systeme gefunden werden, wie zum Beispiel das Civitas System, welches eine große Anzahl an Bedingungen erfüllt. Was im Zusammenhang damit allerdings auch beobachtet werden konnte war, dass die Komplexität des Systems mit der Anzahl der erfüllten Bedingungen steigt.

Der zweite Aspekt dieser Masterarbeit, die restriktive AGHO Blindsignatur [HK19], zeigte eine Möglichkeit auf, wie die Anonymität von privaten Attributen bei einer elektronischen Wahl leicht gewahrt werden kann. Dieses Signaturverfahren wurde für das in dieser Arbeit entstandene Protokoll verwendet. Solch ein Ansatz für das Bewahren der Anonymität ist noch in keinem anderen der beschriebenen Systeme umgesetzt worden.
Die Implementierung der kryptographischen Bibliothek zur Realisierung des Protokolls konnte mit Hilfe von Charm [AMG+11] durchgeführt werden. Die Bibliothek implementierte bereits alle notwendigen Operationen, um mit elliptischen Kurven und Pairings umgehen zu können.
Die entstandene kryptographische Bibliothek enthält mehrere Module. Es ist möglich einfache ElGamal Verschlüsselungen vorzunehmen, die AGHO Signatur Komponenten zu verwenden und auch die dazugehörigen Zero-Knowledge Proofs (User, Server und Auszählbehörde) durchzuführen. All diese Komponenten können mit einer beliebigen Anzahl an Attributen durchgeführt werden, wobei das letzte Attribut als privat angenommen wird.
Die erstellten und durchgeführten Komponententests und auch die prototypische Implementierung eines Client-Server Modells zeigten die Funktionsfähigkeit und die Anwendbarkeit des implementierten Protokolls.
Im Allgemeinen ist dieses Protokoll aufgrund des Aufbaus der AGHO Blindsignatur [HK19] sehr gut für statistische Auswertungen geeignet. Da beliebig viele Attribute in die Stimme hineincodiert werden können, können diese anschließend verwendet werden, um aussagekräftige Statistiken zu erstellen. Die prototypische Implementierung bietet einen guten Einstieg für eine Client-Server Anwendung für die kryptographische Bibliothek. Es gibt viele Aspekte, die durch diese Implementierung beleuchtet wurden, und im weiteren noch behandelt werden können. Sowohl aus IT-Sicherheits-Sicht als auch aus Benutzerfreundlichkeits-Sicht hat diese Implementierung noch sehr viel Entwicklungspotential.
Der anschließende Vergleich hat gezeigt, dass durch die prototypische Implementierung schon einige Bedingungen erfüllt sind. Viele der Bedingungen aus 2.1 sind durch kleine Erweiterun-

gen leicht erfüllbar. Ein wenig aufwändiger wäre es das System als Schwellwert-Kryptosystem zu implementieren und damit die Bedingung der *Fairness* zu erfüllen.
Die Bedingung *Zwang zur Wahl*, welche auch in der Vergangenheit bei vielen Remote Wahlsystemen Probleme gemacht hat, wäre auch hier nur mit großem Aufwand und starkem Verlust der Benutzerfreundlichkeit zu erfüllen. Aufgrund des Aufbaus des Wahlprotokolls ist die Bedingung der *Nicht-Belegbarkeit* nicht erfüllbar, ohne dabei auf eine andere Bedingung verzichten zu müssen. Es hat sich allgemein gezeigt, dass oft für das Erfüllen einer weiteren Bedingung einige andere Bedingungen gelockert oder sogar aufgegeben werden müssen.
Abschließend kann geschlussfolgert werden, dass so wie die in Kapitel 2 vorgestellten elektronischen Wahlsysteme, auch das AGHO Wahlsystem seine Vor- und Nachteile und spezifischen Anwendungsgebiete hat, für die es besonders gut geeignet ist.

Literaturverzeichnis

[Adi08] Ben Adida. Helios: Web-based Open-Audit Voting. *Proceedings of the 17th USENIX Security Symposium*, 01 2008. 11, 16

[Aga17] Sagar Chand Agarwal. Building a Basic RestFul API in Python, 2017. `https://www.codementor.io/@sagaragarwal94/building-a-basic-restful-api-in-python-58k02xsiq`, Zugegriffen am 27.04.2020. 53

[AGHO11] Masayuki Abe, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Optimal Structure-Preserving Signatures in Assymetric Bilinear Groups. *Lecture Notes in Computer Science*, 6841, 2011. 2, 19, 28, 29, 30, 46, 47

[AIK+16] Ioannis Askoxylakis, Sotiris Ioannidis, Sokratis Katsikas, et al. *Computer Security - ESORICS 2016*. LNCS, 2016. 16

[AMG+11] J. Ayo Akinyele, Ian Miers, Christina Garman, et al. Charm: A Framework for Rapidly Prototyping Cryptosystems, 2011. `https://github.com/JHUISI/charm`, Zugegriffen am 18.04.2020. 45, 53, 64

[Aut] Microsoft Authenticator. `https://support.microsoft.com/de-at/help/4026727/`, Zugegriffen am 29.04.2020. 63

[Aye17] Ahmed Ben Ayed. A Conceptual Secure Blockchain- Based Electronic Voting System. *International Journal of Network Security & Its Applications (IJNSA)*, 9(3), 2017. 17

[BBB+13] Susan Bell, Josh Benaloh, Michael D. Bryne, et al. STAR-Vote: A Secure, Transparent, Auditable, and Reliable Voting System. *USENIX Journal of Election Technology and Systems*, 1(1), 08 2013. 6, 7, 16, 69

[BBH+17] Matthew Bernhard, Josh Benaloh, J. Alex Halderman, et al. Public Evidence from Secret Ballots. *CoRR*, abs/1707.08619, 2017. 1, 4, 5, 6, 7, 10, 11, 16, 17

[BDS17] Alessandro Bruni, Eva Drewsen, and Carsten Schürmann. Towards a Mechanized Proof of Selene Receipt-Freeness and Vote-Privacy. *Electronic Voting*, pages 110–126, 01 2017. 11, 16

[Ber01] HU Berlin. Moderne Verfahren der Kryptographie, 2001. `https://www2.informatik.hu-berlin.de/Forschung_Lehre/algorithmenII/Lehre/WS2001-2002/Krypto/rsa.htm`, Zugegriffen am 30.03.2020. 19

[Ber02] HU Berlin. ElGamal, 2002. `https://www2.informatik.hu-berlin.de/Forschung_Lehre/algorithmenII/Lehre/SS2002/Ana_krypt_Alg/10EC/html/node15.html`, Zugegriffen am 30.03.2020. 20, 21

[BKV17] David Bernhard, Oksana Kulyk, and Melanie Volkamer. Security Proofs for Participation Privacy, Receipt-Freeness and Ballot Privacy for the Helios Voting Scheme. *12th International Conference*, 2017. 16

[CCC+08a] David Chaum, Richard Carback, Jeremy Clark, et al. Scantegrity II: End-to-End Verifiability for Optical Scan Election Systems using Invisible Ink Confirmation Codes. *USENIX/ACCURATE Electronic Voting Workshop*, 2008. 5, 6, 69

[CCC+08b] David Chaum, Richard T. Carback, Jeremy Clark, et al. Scantegrity, 2008. http://scantegrity.com, Zugegriffen am 10.09.2019. 1

[CCM08] Michael R. Clarkson, Stephen Chong, and Andrew C. Myers. Civitas: Toward a Secure Voting System. *2008 IEEE Symposium on Security and Privacy*, 2008. 1, 13, 14, 16, 69

[CEC+08] David Chaum, Aleks Essex, Richard Carback, et al. Scantegrity: End-to-End Voter-Verifiable Optical-Scan Voting. *IEEE Security and Privacy Magazine*, 6(3):40–46, 06 2008. 1, 16

[Cos17] Craig Costello. Pairings for beginners. 2017. http://www.craigcostello.com.au/pairings/PairingsForBeginners.pdf, Zugegriffen am 20.04.2020. 29

[CP92] David Chaum and Torben P. Pedersen. Wallet Databases with Observers. *Advances in Cryptology — CRYPTO' 92*, pages 89–105, 1992. 11, 12

[Dre] TU Dresden. Zero-Knowledge-Verfahren. https://tu-dresden.de/ing/informatik/sya/ps/ressourcen/dateien/studium/materialien/mat_kp_datensicherheit/v11_doku.pdf?langde, Zugegriffen am 29.02.2020. 23, 24, 26, 27, 28, 69

[EE19] E-Estonia. i-voting, 2019. https://e-estonia.com/solutions/e-governance/i-voting/, Zugegriffen am 10.9.2019. 17

[Gre] Matthew Green. A Few Thoughts on Cryptographic Engineering. https://blog.cryptographyengineering.com/a-note-on-blind-signature-schemes/, Zugegriffen am 20.02.2020. 20

[Gre14] Matthew Green. Zero Knowledge Proofs: An illustrated primer, 2014. https://blog.cryptographyengineering.com/2014/11/27/zero-knowledge-proofs-illustrated-primer/, Zugegriffen am 09.04.2020. 25, 26, 69

[Hao17] Edward Hao. RFC 8235: Schnorr Non-interactive Zero-Knowledge Proof. *RFC*, 2017. 28

[HK19] Ulrich Haböck and Stephan Krenn. Breaking and Fixing Anonymous Credentials for the Cloud. *Cryptology and Network Security*, pages 249–269, 2019. 2, 30, 31, 32, 34, 38, 40, 46, 47, 64, 69

[HL] Markus Hermann and Dejan Lazich. Kryptographie mit elliptischen Kurven. https://crypto.iti.kit.edu/fileadmin/User/Embedded_Security_SS10/4-EC-Krypto.pdf, Zugegriffen am 20.02.2020. 21

[JKK18] Wojciech Jamroga, Michal Knapik, and Damian Kurpiewski. Model Checking the SELENE E-Voting Protocol in Multi-agent Logics. *Electronic Voting*, pages 100–116, 01 2018. 16

[Kam15] Soham Kamani. How to communicate between Python and NodeJS, 2015. https://www.sohamkamani.com/blog/2015/08/21/python-nodejs-comm/, Zugegriffen am 20.04.2020. 63

[Kap20] Andrea Kapsch. AGHO Blind kryptlib, 2020. https://github.com/AndreaKap/AGHO_Blind_kryptlib.git, Zugegriffen am 08.05.2020. 45

[KSM+16] Ralf Küsters, Enrico Scapin, Johannes Muller, et al. sElect: A Lightweight Verifiable Remote Voting System. *IEEE 29th Computer Security Foundations Symposium*, 06 2016. 1, 12, 13, 16

[Luc] Lucidchart. `https://www.lucidchart.com/`, Zugegriffen am 29.04.2020. 53, 69

[Lyn] Ben Lynn. PBC Library. `https://crypto.stanford.edu/pbc/`, Zugegriffen am 08.05.2020. 45

[oMH13] Broad Institute of MIT and Harvard. Calling Python from Java, 2013. `https://pythonhosted.org/javabridge/java2python.html`, Zugegriffen am 20.04.2020. 63

[RBHS10] Peter Y. A. Ryan, David Bismark, James Heather, and Steve Scheider. Prêt à Voter: a Voter-Verifiable Voting System. *IEEE Transactions on Information Forensics and Security*, 4(4):662–673, 01 2010. 4, 5, 16, 69

[Riv06] Ronald L. Rivest. The ThreeBallot Voting System. 10 2006. 1, 7, 8, 16, 69

[RRI16] Peter Y. A. Ryan, Peter B. Ronne, and Vincenzo Iovino. Selene: Voting with Transparent Verifiability and Coercion-Mitigation. *Financial Cryptography and Data Security*, pages 176–192, 08 2016. 11, 16

[RS07] Ronald L. Rivest and Warren D. Smith. Three Voting Protocols: ThreeBallot, VAV, and Twin. 08 2007. 8, 9, 16, 69

[SK95] Kazue Sako and Joe Kilian. Receipt-Free Mix-Type Voting Scheme - A practical solution to the implementation of a voting booth. *EUROCRYPT'95: Proceedings of the 14th annual international conference on Theory and application of cryptographic techniques*, pages 393–403, 1995. 11

[SMH17] Alexander Schneider, Christian Meter, and Philipp Hagemeister. Survey on remote electronic voting. *CoRR*, abs/1702.02798, 2017. 3, 13, 16, 17, 18

[WWI+12] Scott Wolchok, Eric Wustrow, Dawn Isabel, et al. Attacking the Washington, D.C. Internet Voting System. *Financial Cryptography and Data Security*, pages 114–128, 2012. 18

[YCK+19] Shoko Yonezawa, Sakae Chikara, Tetsutaro Kobayashi, et al. Pairing-Friendly Curves, 2019. `https://tools.ietf.org/id/draft-yonezawa-pairing-friendly-curves-00.html`, Zugegriffen am 08.05.2020. 46

[ZCC+13] Filip Zagorski, Richard Carback, David Chaum, et al. Remotegrity: Design and Use of an End-to-End Verifiable Remote Voting System. *ACNS*, pages 441–457, 2013. 1, 9, 10, 16, 69

Abbildungsverzeichnis

Tabellenverzeichnis

Listings