Karlsruhe Institute of Technology

Skript zur Vorlesung
Kryptographische Wahlverfahren

Sommersemester 2013

Institut für Kryptographie und Sicherheit
Europäisches Institut für Systemsicherheit
Fakultät für Informatik
Karlsruher Institut für Technologie

Autoren: Bernhard Löwe, Carmen Kempka

Version: 8. Oktober 2013

IKS Institut für Kryptographie und Sicherheit

EISS

Europäisches Institut für Systemsicherheit
Institut für Kryptographie und Sicherheit
Fakultät für Informatik
Karlsruher Institut für Technologie
Am Fasanengarten 5
76128 Karlsruhe

Vorwort

Wahlmaschinen und elektronisch unterstützte Wahlen gewinnen weltweit immer mehr an Bedeutung. In den USA und auch in Europa werden einige Parlamentswahlen elektronisch unterstützt und teilweise sogar über das Internet durchgeführt. Wahlmaschinen versprechen eine schnellere Auszählung, die bequeme Internetwahl eine höhere Wahlbeteiligung. Andererseits sind die damit einhergehenden Sicherheitsprobleme und Gefahren immer wieder Diskussionsgegenstand.

Das Ziel kryptographischer Wahlverfahren ist, dem Wähler zu ermöglichen, seine eigene Stimmabgabe sowie die Korrektheit der Auszählung zu verifizieren, selbst wenn die Stimme an einem korrumpierten Gerät abgegeben wurde. Dabei soll das Wahlgeheimnis selbverständlich gewahrt bleiben, und auch die Benutzerfreundlichkeit darf nicht unter dem Einsatz der Kryptographie leiden.

Dieses Skript entsteht parallel zur Vorlesung Kryptographische Wahlverfahren im Sommersemester 2013, ist entsprechend noch unvollständig und wird fortlaufend während der Laufzeit der Vorlesung auf der Vorlesungshomepage aktualisiert werden.

Die Vorlesung richtet sich an Studenten der Informatik im Masterstudium. Vorkenntnisse, die den Inhalt der im Bachelor angebotenen Informatikvorlesungen übersteigen, sind nicht erforderlich. Eine gewisse Vertrautheit mit Grundbegriffen und Mechanismen der Kryptographie ist aber hilfreich.

Inhaltsverzeichnis

Kapitel 1
Einleitung

Bei der klassischen Papierwahl, wie sie in Deutschland für politische Wahlen verwendet wird, wird dem Wähler die Möglichkeit gegeben, durch ein Kreuz auf einem Stimmzettel, den er anschließend in eine Urne wirft, seine Stimme abzugeben. Der Wähler kann sich selbst davon überzeugen, dass sein Stimmzettel in der Urne gelandet ist - er hat ihn schließlich selbst hineingeworfen - und durch Wahlbeobachtung kann sichergestellt werden, dass das Wahlergebnis richtig ausgezählt wurde. Wird nun diese Art der Stimmabgabe durch einen Wahlcomputer ersetzt oder verläuft sogar dezentral über das Internet, ist diese Nachvollziehbarkeit so nicht mehr gegeben, es müssen neue Maßnahmen zum Erhalt der verifizierbaren Korrektheit der Wahl herangezogen werden.

Kryptographische Wahlverfahren sind Wahlverfahren, die eine solche Überprüfbarkeit der Korrektheit des Wahlergebnisses ermöglichen, ohne dabei das Wahlgeheimnis zu verletzen. Diese Überprüfbarkeit soll natürlich auch dann noch gegeben sein, wenn die Stimme an einer manipulierten Wahlmaschine oder über das Internet am eigenen, evtl. virenbefallenen Rechner abgegeben wurde.

Dabei geht es nicht immer nur um politische Wahlen: Auch für Vorstandswahlen in Vereinen, Abstimmungen in Online-Communities u.s.w. können kryptographische Wahlverfahren interessant sein, und sei es nur, um die Integrität der Wahl bei evtl. fehlerhafter Software sicherzustellen, ohne gleich offen zu wählen.

Auch beschäftigen wir uns nicht ausschließlich mit Wahlmaschinen. In der Vorlesung werden wir verschiedene Typen von Wahlverfahren kennenlernen. Zum einen gibt es *papierbasierte Verfahren*, bei denen der Wähler seine Stimme durch Ausfüllen und einwerfen (bzw. einscannen lassen) eines Papierstimmzettels abgibt, wobei er selbst nicht in Berührung mit Computern kommt, die Stimme jedoch unter Umständen durchaus computergestützt weiterverarbeitet werden kann.

Bei *elektronisch unterstützten Wahlverfahren* gibt der Wähler seine Stimme an einem Wahlcomputer oder einer sogenannten *DRE (direct-recording electronic)* ab. Eine DRE ist eine Art Wahlmaschine, die die abgegebenen Stimmen elektronisch auf einem herausnehmbaren Speicher abspeichert. Oft sprechen wir auch hier von einem “Stimmzettel", der in irgendeiner Form die Wählerstimme widerspiegelt, von dem jedoch evtl. nur eine elektronische Represäntation existiert. Vor allem bei nicht kryptographischen Wahlen wird oft zusätzlich ein sogenanntes *VVPAT (voter verifiable paper audit trail)* ausgedruckt, das im wesentlichen aus den ausgefüllten Stimmzetteln besteht.

Des weiteren unterscheiden wir zwischen Präsenzwahlen und Internet- oder Onlinewahlen. Bei der Präsenzwahl gibt der Wähler seine Stimme in einem Wahlbüro ab, je nach Verfahren auf einem Papierstimmzettel oder an einem Wahlcomputer. Bei der Internetwahl muss sich der Wähler nicht mehr zu einem Wahlbüro begeben, sondern kann seine Stimme an einem beliebigen Computer, der über eine Verbindung zum Internet verfügt, abgeben.

Neben verschiedenen Wahlverfahren und den zum Verständnis erforderlichen kryptographischen Grundlagen beschäftigen wir uns außerdem mit Sicherheitsdefinitionen von Wahlverfahren. Dabei werden aus zwei Modellen die jeweiligen Definitionen der *Nicht-Erpressbarkeit*, einer stärkeren Form des Wahlgeheimnisses, vorgestellt.

Zu guter Letzt werden wir auch einige hierzulande weniger bekannten Wahlformen und Sonderfälle wie die Präferenzwahl, die Wahl mit write-in-Kandidaten sowie delegated voting (Wahl mit Möglichkeit zur Stimmweitergabe) behandeln.

1.1 Was sind kryptographische Wahlverfahren?

Von kryptographischen Wahlen sprechen wir dann, wenn i. d. R. mathematische Methoden verwendet werden, um die Anforderungen, die an eine Wahl gestellt werden (mehr dazu in Kapitel 1.2), sicher zu stellen.

Der Wahlvorgang, wie wir ihn kennen, enthält drei wesentliche Schritte:

1. Der Wähler weist seine Wahlberechtigung nach

2. Der Wähler gibt seine Stimme ab
3. Die Stimmen werden ausgezählt

Dabei soll das Wahlgeheimnis geschützt bleiben, und jeder soll nachprüfen können, ob das Wahlergebnis korrekt berechnet wurde. Unsere Papierwahl erfüllt diese Anforderungen weitestgehend durch den Einsatz von Wahlkabinen und Urnen und durch die Möglichkeit, die Wahl und die Auszählung zu beobachten.

Mit Wahlmaschinen und Internetwahlen wird es etwas komplizierter. Die Stimmabgabe wird bei Einsatz von Wahlcomputern an einem nicht unbedingt vertrauenswürdigen Rechner durchgeführt, der dann auch für die korrekte Speicherung und ggf. Löschung von Stimmabgabe-Daten verantwortlich ist. Bei einer Internetwahl wird die Idendität eines Wählers und seine Wahlberechtigung nicht mehr persönlich von Menschen vor Ort im Wahllokal geprüft, sondern ist das Ergebnis einer Interaktion zwischen dem vom Wähler verwendeten Rechner und mindestens einem Wahlserver. Bei der Internetwahl wird der einfach zu überblickende Vorgang des Einwurfs des Stimmzettels in die Wahlurne durch einen Vorgang ersetzt, an dem sowohl der Rechner des Wählers als auch mindestens ein Wahlserver[1] beteiligt ist. Der Wählerrechner muss die Stimme richtig verarbeiten und weiterleiten, der Wahlserver muss die Stimme richtig empfangen und abspeichern, und bei der Auszählung müssen genau diese Stimmen richtig gezählt werden. Die Beteiligung von privaten Rechnern und Servern, deren Betreiber und Administratoren nicht per Definition vertrauenswürdig sein können, macht eine von Rechnern und Implementierungen unabhängige Verifizierbarkeit wünschenswert wenn nicht sogar notwendig. Das Ziel kryptographischer Wahlverfahren ist die Verifizierbarkeit der Korrektheit des Wahlergebnisses, unabhängig von der Implementierung von Wahlservern oder des Rechners, an dem die Stimme abgegeben wurde. Das Wahlgeheimnis muss dabei gewahrt bleiben.

1.1.1 Warum Wahlcomputer - Vorteile von elektronischen Wahlen gegenüber der Papierwahl

Trotz der vielen Diskussionen über die Nachteile von Wahlmaschinen sind elektronisch unterstützte Wahlverfahren bereits in vielen Teilen der Erde im Einsatz. In den USA wird die Stimmauszählung schon seit längerem durch die Stimmabgabe am Wahlcomputer oder scannerbasierte Systeme unterstützt. In einigen Teilen Europas wie Norwegen und Estland können Wähler bereits in Parlamentswahlen ihre Stimme über das Internet abgeben. Elektronische Wahlverfahren versprechen einige Vorteile:

- Effizientere Auszählung
- Weniger Papierverbrauch
- Genauere Auszählung durch Vermeidung menschlicher Fehler
- Unterstützung des Wählers beim Ausfüllen des Wahlzettels, dadurch
- kein versehentliches Abgeben eines ungültigen Stimmzettels
- Internetwahl möglich
  - Wahl bequem von zu Hause aus möglich, dadurch
  - höhere Wahlbeteiligung

1.1.2 Probleme von Wahlcomputern - Warum kryptographische Wahlverfahren?

Die Auszählung ist nur dann genauer oder überhaupt korrekt, wenn der Wahlcomputer keine Programmierfehler enthält. Die Korrektheit von Software zu überprüfen ist schwer, theoretisch sogar unmöglich, da das Halteproblem darauf reduziert werden kann. Schon allein dadurch erschwert der Einsatz von Computern nicht nur dem Laien die Nachvollziehbarkeit. In den Medien hat sich hier der Begriff *Black Box Voting* etabliert, der verdeutlicht, wie schwer nachvollziehbar es ist, was

[1] Unter einem Wahlserver ist ein Computer zu verstehen, der zentraler Bestandteil aller Wahlvorgänge ist. Im Vergleich dazu ist der Wählerrechner i. All. nur an einer oder wenigen Wahlen beteiligt, steht vor Beginn der Wahl noch nicht als beteiligte Komponente fest und erfüllt keine spezialisierte Aufgabe.

mit der Stimme nach der Eingabe in die Wahlmaschine passiert, wie und ob überhaupt sie gezählt wurde. Dem entgegen treten kryptographische Wahlverfahren mit der Eigenschaft der *Software Independence*. Das bedeutet, dass die Korrektheit des Ergebnisses unabhängig von der Implementierung des Wahlcomputers verifizierbar ist. Selbst bei einem korrumpierten Wahlcomputer, auf dem beliebige Software läuft, können Manipulationen immer oder zumindest mit ausreichend hoher Wahrscheinlichkeit entdeckt werden.

1.2 Anforderungen an Wahlverfahren

In diesem Abschnitt sollen einige Anforderungen an Wahlverfahren identifiziert werden. Es ist wünschenswert, dass kryptographische Wahlverfahren mindestens das gleiche leisten können wie bisher eingesetzte Wahlverfahren. Deshalb werden wir uns im weiteren Verlauf dieses Kapitels die Papierwahl in Deutschland und die deutsche Wahlordnung als Beispiel ansehen. In den nächsten beiden Abschnitten werden jedoch zunächst einmal einige mögliche Angriffe vorgestellt und daraufhin ein paar wünschenswerte Eigenschaften von Wahlverfahren gesammelt. Einige dieser Eigenschaften werden im späteren Verlauf der Vorlesung (Thema “Sicherheitsdefinitionen”, Kapitel 4) noch genauer definiert.

1.2.1 Angriffe auf Wahlverfahren

Bevor wir die wichtigsten Anforderungen an Wahlverfahren vorstellen, betrachten wir zunächst einige Angriffe, vor denen kryptographische Wahlverfahren schützen wollen:

- **Stimmkauf:** Der Wähler bekommt Geld für die Abgabe einer bestimmten Stimme oder eine der bei Wählererpressung aufgelisteten Aktionen (siehe nächster Punkt) geboten.
- **Wählererpressung:** Der Wähler wird in irgendeiner Form dazu erpresst, nicht das zu wählen, was er eigentlich wollte. Dies kann auf unterschiedliche Art[2] passieren:
  - Der Wähler wird gezwungen, einen bestimmten Kandidaten zu wählen.
  - **Erzwungene Enthaltung:** Der Wähler wird gezwungen, nicht zur Wahl zu gehen.
  - **Randomisierungsangriff:** Der Wähler wird gezwungen, eine zufällige Stimme abzugeben, die mit hoher Wahrscheinlichkeit nicht dem Kandidaten entspricht, den er eigentlich wählen wollte[3].
  - **Simulationsangriff:** Der Wähler wird gezwungen, seine Zugangsdaten für die (online-) Wahl herauszugeben.
  - **Chain Voting:** Der Wähler muss einen vom Angreifer erhaltenen und bereits ausgefüllten Stimmzettel in die Urne im Wahllokal einwerfen und einen leeren Stimmzettel wieder mit herausbringen, den der Angreifer ausfüllen und dem nächsten Wähler geben kann. Diesen Vorgang kann der Angreifer wiederholen, da im nun wieder ein leerer Stimmzettel zur Verfügung steht.
  - **Pattern Voting:** Der Wähler wird gezwungen, seinen Wahlzettel in bestimmtem Muster auszufüllen, in das er seine Idendität codiert und damit sein Wahlgeheimnis aufgibt.
- **Manipulation der Stimmen:** Der Angreifer versucht, die Integrität der Wahl zu brechen. Dies kann er auf unterschiedliche Weise tun:
  - Der Angreifer manipuliert einzelne Wählerstimmen.
  - Der Angreifer entfernt abgegebene Stimmen.
  - Der Angreifer tauscht abgegebene Stimmen durch eigene aus.

[2] Einige der Angriffe werden in [JCJ05] vorgestellt.

[3] Um den Angriff zu verdeutlichen, stellen wir uns als Beispiel ein einfaches Wahlverfahren vor, in dem der Wähler einen Wahlzettel bekommt, auf dem sich auf der linken Seite die Kandidaten in zufällig permutierter Reihenfolge, und auf der rechten Seite Kästchen zum Ankreuzen und die verschlüsselte Permutation befinden. Der Wähler macht sein Kreuz, die rechte Hälfte dient zum Auszählen und eine Kopie davon als Quittung, die linke wird vernichtet. Der rechten Hälfte sieht man nun zwar nicht an, was der Wähler gewählt hat, der Angreifer kann jedoch den Wähler zwingen, das obere Kästchen anzukreuzen, egal, für welchen Kandidaten es steht, und so mit hoher Wahrscheinlichkeit verhindern, dass der Wähler seinen gewünschten Kandidaten wählt. Das oben beschriebene Beispielwahlverfahren ist angelehnt an das Verfahren Prêt à Voter (http://www.pretavoter.com/, in dieser Vorlesung nicht behandelt).

  - **Ballot Stuffing:** Es werden unberechtigterweise Stimmen dazugemogelt, z.B. von einem Wahlserver.

- **Unberechtigtes Wählen:**
  - Ein Wähler gibt mehr als einen Stimmzettel ab.
  - Ein Wähler gibt eine Stimme ab, obwohl er nicht wahlberechtigt ist.

- **Abbrechen der Wahl:** Der Angreifer täuscht eine Manipulation vor, die prinzipiell nicht bewiesen werden kann, und erzwingt so eine Wahlwiederholung.

Viele dieser Angriffe und der Anforderungen müssen nicht durch das kryptographische Wahlverfahren abgedeckt werden, sondern können organisatorisch gelöst werden. Chain Voting lässt sich beispielsweise vermeiden, indem jeder so viele Wahlzettel bekommt, wie er will, aber nur einen einwerfen darf. Auch Pattern Voting lässt sich in vielen Fällen organisatorisch vermeiden, z.B. durch sogenanntes *contest partitioning* [PS07], d.h. wenn ein Wähler mehr als eine Stimme zur Verfügung hat, wodurch er Information in seine Stimmabgabe codieren und damit seinen Stimmzettel wiedererkennbar machen könnte, so wird die Wahl aufgesplittet in mehrere Wahlen (oder einfach mehrere Stimmzettel), so dass jeder Wähler pro Wahlvorgang bzw. Stimmzettel nur noch eine Stimme abgeben kann.

1.2.2 Anforderungen an Wahlverfahren

Nachdem wir nun einige mögliche Angriffe kennengelernt haben, fassen wir hier einige Anforderungen an Wahlverfahren zusammen. Sie sind grob anhand der Wahlgrundsätze in Deutschland strukturiert, eine genaue Einteilung läßt sich jedoch nicht vornehmen.

Nicht jedes der Wahlverfahren, die wir hier vorstellen, setzt alle dieser Anforderungen um, oft ist dies auch nicht notwendig, unterschiedliche Wahlen haben unterschiedliche Anforderungen, außerdem lassen sich, wie oben schon angedeutet, einige Angriffe auch organisatorisch verhindern.

Anforderungen, die die *geheime, freie* Wahl betreffen:

- **Wahlgeheimnis:** Während und nach der Wahl darf keine Information bekannt werden, die darauf schließen lässt, was ein Wähler gewählt hat.

- **Quittungsfreiheit/Belegfreiheit:** Selbst, wenn der Wähler es wollte, könnte er einem Dritten nicht beweisen[4], was er gewählt hat.

- **Nicht-Erpressbarkeit/Unmöglichkeit von Stimmkauf:** Der Wähler kann, selbst wenn er es wollte, nicht in einer Art und Weise seine Stimme abgeben, die dem Angreifer beweisen könnte, dass der Wähler mit hoher Wahrscheinlichkeit nicht das gewählt hat, was er eigentlich vorhatte [5].

- **Keine erzwungene Enthaltung:** Eine wahlberechtigte Person kann nicht gezwungen oder erpresst werden, nicht zu wählen. Diese Anforderung gehört streng genommen zur Nicht-Erpressbarkeit, wird aber oft als Sonderfall behandelt, da diese Anforderung z.B. bei Präsenzwahlen prinzipiell unmöglich[6] zu erreichen ist.

[4] Quittungsfreiheit heißt entgegen der Bezeichnung *nicht*, dass der Wähler keine Quittung in irgendeiner Form bekommt. Er kann durchaus eine Quittung bekommen, mit der er seine Stimmabgabe überprüfen kann. Er darf mit einer eventuellen Quittung nur nicht einem dritten *beweisen* können, für welchen Kandidaten er gestimmt hat.

[5] Der Unterschied zur Quittungsfreiheit ist, dass hier der Fall betrachtet wird, dass der Angreifer den Wähler beeinflussen könnte, ohne die tatsächlich abgegebene Stimme zu lernen. Angenommen, ein Wahlverfahren ist Quittungsfrei und der Angreifer kennt die Vorlieben eines bestimmten Wählers. Nun könnte er aber evtl. immer noch, ohne die tatsächlich abgegebene Stimme zu lernen, zumindest versuchen, einem Kandidaten mit hoher Wahrscheinlichkeit eine Stimme wegzunehmen. Wir erinnern nochmal an das Beispiel Prêt à Voter aus einer früheren Fußnote: Das Verfahren ist quittungsfrei, es ist aber ein Ransomisierungsangriff möglich. Wenn der Wähler jedoch in einem Wahlverfahren kein bestimmtes Verhalten nachweisen kann, das die Stimme beeinflusst hätte (wie z.B. ein bestimmtes Kästchen anzukreuzen), weiß der Stimmkäufer nicht, ob es sich lohnt zu zahlen, und der Erpresser nicht, ob die Erpressung erfolgreich war.

[6] Man kann immer das Wahllokal beobachten und schauen, wer zur Wahl geht. Außerdem ist manchmal das Veröffentlichen von Wählerlisten notwendig, z.B. um sogenanntes *Ballot Stuffing* (dazu später mehr) zu verhindern.

- **Keine vorläufigen Zwischenergebnisse:** Vorläufige Zwischenergebnisse lassen die Auswahl eines Wählers, der in einem bestimmten Zeitraum gewählt hat, mit höherer Wahrscheinlichkeit[7] bestimmen, als wenn nur das Gesamtergebnis vorliegt. Diese Anforderung gehört aus einem anderen Grund außerdem auch zur *Gleichheit* der Wahl.

Anforderungen, die die *Korrektheit* und die *Nachvollziehbarkeit* bzw. *Öffentlichkeit* der Wahl betreffen:

- **Korrektheit:** Das Wahlergebnis wird aus den abgegebenen Stimmen korrekt berechnet.
- **Integrität:** Es werden genau die Stimmen gezählt, die von wahlberechtigten Wählern abgegeben wurden. Es werden keine Stimmen unbefugt hinzugefügt, gelöscht oder verändert.
- **Verifizierbarkeit:** Jeder kann nachprüfen, ob die eigene Stimme in der Auszählung berücksichtigt wurde (*individuelle Verifizierbarkeit*) und ob das Wahlergebnis korrekt berechnet wurde (*universelle Verifizierbarkeit*)

Anforderungen, die die *Gleichheit* bzw. *Fairness* der Wahl betreffen:

- **Gleiches Stimmgewicht:** Jeder Wähler kann nur eine Stimme abgeben und jede Stimme wird nur einmal gezählt. (Wir werden später Verfahren kennenlernen, die die *Stimmdelegation*, d.h. das Weitergeben des eigenen Stimmgewichts an eine Vertrauensperson, oder sogenanntes *Revoting*, also das Überschreiben der eigenen, bereits abgegebenen Stimme mit einer Neuen, erlauben. Diese Anforderung müsste in diesen beiden Fällen natürlich in ihrer Formulierung entsprechend angepasst werden, sinngemäß gilt sie aber weiterhin.)
- **Keine vorläufigen Zwischenergebnisse:** Jeder soll unter gleichen Voraussetzungen, also auch mit der gleichen Information zur Stimmabgabe gehen. Diese Anforderung gehört außerdem auch zum *Wahlgeheimnis*.

Sonstige wünschenswerte oder möglicherweise notwendige Anforderungen:

- **Robustheit:** Niemand soll einen Abbruch der Wahl oder eine Wiederholungswahl provozieren können. Dazu muss eine Manipulation beweisbar sein, damit zwischen einer echten und einer behaupteten Manipulation unterschieden werden kann.
- **Benutzerfreundlichkeit:** intuitiv, leicht benutzbar, schnell, ...
- Möglichkeit einer **späteren Nachzählung / Archivierung** der Stimmen
- **Flexibilität:** z.B. Möglichkeit von Kumulieren, Panaschieren, Write-In-Kandidaten, ...
- **Korrigierbarkeit:** Möglichkeit, im Nachhinein als unberechtigt/falsch abgegeben identifizierte Stimme zu löschen oder zu ändern)
- **Wahlspezifische Zusatzanforderungen:** z.B. vorgegebene Reihenfolge der Kandidaten auf Stimmzettel, vorgegebenes Layout, ungültig wählen möglich, ...

[7] Wird ein Zwischenergebnis über k Stimmen bekannt, so ist jede der ins Zwischenergebnis eingeflossenen Stimmen eben nicht nur eine aus n Stimmen (n Anzahl Wähler), sondern eine aus diesen (wenigeren!) k Stimmen, eine Stimme lässt sich nun also mindestens mit Wahrscheinlichkeit $\frac{1}{k}$ raten, statt mit der kleineren Wahrscheinlichkeit $\frac{1}{n}$. Im Extremfall $k = 1$ (oder $k = 2$ und man hat selbst die andere Stimme abgegeben) wäre für diese Stimme das Wahlgeheimnis gebrochen. Je mehr Stimmen in ein Ergebnis einfließen, umso weniger läßt sich vom Wahlergebnis auf bestimmte einzelne Stimmen schließen.

1.2.3 Die Papierwahl in Deutschland als Beispiel

Die Eigenschaften, wie sie von elektronischen Wahlverfahren gefordert werden, müssen in der ein oder anderen Variante auch von den Systemen, wie sie derzeit für die Bundestagswahl angewendet werden, garantiert werden können. Statt, wie es bei Onlinewahlverfahren notwendig ist, sich vollständig auf die Architektur und die verwendeten kryptographischen Verfahren zu verlassen, wird die Sicherheit bei der Papierwahl durch den Prozess, der in der Bundeswahlordnung beschrieben ist, gewährleistet. Wie in jedem Verfahren müssen auch hier Kompromisse zwischen Sicherheit und Anwendbarkeit bzw. Verfügbarkeit eingegangen werden. So können beispielsweise mit der Briefwahl einige der Mechanismen umgangen werden, die gewährleisten, dass sich der Wähler frei entscheiden kann. In diesem Kapitel wird die Bundeswahlordnung [bwo12] und das Bundeswahlgesetz [bwg13] auszugsweise vorgestellt und erläutert, welche sicherheitsrelevanten Eigenschaften für den Wahlvorgang daraus folgen, sofern er wie beschrieben durchgeführt wird.

1.2.3.1 Öffentliche Verifizierbarkeit des Wahlergebnisses

Paragraph 54 garantiert jedem, egal ob wahlberechtigt oder nicht, den Zutritt zum Wahlraum, wo er den Vorgang der Stimmabgabe und der Stimmauszählung beobachten kann:

> **BWO §54 Öffentlichkeit**
> Während der Wahlhandlung sowie der Ermittlung und Feststellung des Wahlergebnisses hat jedermann zum Wahlraum Zutritt, soweit das ohne Störung des Wahlgeschäfts möglich ist.

Da es nicht nur ein Wahllokal geben kann, ist es einer einzelnen Person bei einer Bundestagswahl jedoch nicht möglich den ganzen Wahlvorgang zu beobachten. Dazu müsste er sich gleichzeitig in jedem einzelnen Wahllokal befinden. Die Möglichkeit einer öffentlichen Kontrolle der Korrektheit des Wahlergebnisses ist also nur dadurch gegeben, dass Teilergebnisse kontrolliert werden können. Eine Gruppe sich gegenseitig vertrauender Einzelpersonen kann selbstverständlich die Zahl der kontrollierten Teilergebnisse beliebig, bis hin zu einer vollständigen Abdeckung, erhöhen.

1.2.3.2 Frei und geheim

Als frei wird eine Wahl dann bezeichnet, wenn der Wähler seine Stimme selbst und unbeeinflusst abgeben kann. Unbeeinflusst bedeutet insbesondere auch, dass er keinem Zwang unterliegen darf, welche Wahl er zu treffen hat. Das Wahlgeheimnis hingegen garantiert dem Wähler, dass seine Wahl auch nach dem Wahlvorgang nicht bekannt wird. Diese zwei Anforderungen liegen sehr eng bei einander und die Maßnahmen, die diese Garantien sicher stellen sollen, überschneiden sich daher.

Sowohl bei herkömmlichen Papierwahlen als auch bei elektronischen Wahlverfahren und erst recht bei Onlinewahlverfahren sind diese zwei Anforderungen nicht einfach umzusetzen. Wird bei Wahlen die Stimme in einem Wahllokal abgegeben, kann zumindest mit einfachen Mitteln sicher gestellt werden, dass der Wähler seine Wahl unbeobachtet kennzeichnen[8] kann. Das alleine ist allerdings nicht ausreichend, um die Freiheit und Geheimheit der Wahl zu garantieren. Im Folgenden werden verschiedene Paragraphen genannt, mit deren Hilfe diese Ziele erreicht werden sollen.

> **BWO §50 Wahlzelle**
> (1) In jedem Wahlraum richtet die Gemeindebehörde eine Wahlzelle oder mehrere Wahlzellen mit Tischen ein, in denen der Wähler seinen Stimmzettel unbeobachtet kennzeichnen und falten kann. Die Wahlzellen müssen vom Tisch des Wahlvorstandes aus überblickt werden können. Als Wahlzelle kann auch ein nur durch den Wahlraum zugänglicher Nebenraum dienen, wenn dessen Eingang vom Tisch des Wahlvorstandes aus überblickt werden kann.
> (2) In der Wahlzelle soll ein Schreibstift bereitliegen.

Die Wahlzelle ermöglicht es dem Wähler seinen Stimmzettel unbeobachtet zu kennzeichnen. Der Wahlvorstand hat sicher zu stellen, dass sich der Wähler alleine in der Wahlzelle befindet. Ausnahmen werden in §57 geregelt.

> **BWO §51 Wahlurnen**
> (1) Die Gemeindebehörde sorgt für die erforderlichen Wahlurnen.

[8] Setzen des Kreuzes

(2) Die Wahlurne muss mit einem Deckel versehen sein. Ihre innere Höhe soll in der Regel 90 cm, der Abstand jeder Wand von der gegenüberliegenden mindestens 35 cm betragen. Im Deckel muss die Wahlurne einen Spalt haben, der nicht weiter als 2 cm sein darf. Sie muss verschließbar sein.

Durch die Mindestgröße der Urne wird sichergestellt, dass die Einwurfreihenfolge der Stimmzettel später nicht mehr ersichtlich ist, und die Stimmzettel damit nicht anhand dieser Reihenfolge ihren Wählern zugeordnet werden können.

BWO §56 Stimmabgabe
[...]
(2) Der Wähler begibt sich in die Wahlzelle, kennzeichnet dort seinen Stimmzettel und faltet ihn dort in der Weise, dass seine Stimmabgabe nicht erkennbar ist. Der Wahlvorstand achtet darauf, dass sich immer nur ein Wähler und dieser nur so lange wie notwendig in der Wahlzelle aufhält.
[...]
(6) Der Wahlvorstand hat einen Wähler zurückzuweisen, der [...]
4. seinen Stimmzettel außerhalb der Wahlzelle gekennzeichnet oder gefaltet hat oder
5. seinen Stimmzettel so gefaltet hat, dass seine Stimmabgabe erkennbar ist, oder ihn mit einem äußerlich sichtbaren, das Wahlgeheimnis offensichtlich gefährdenden Kennzeichen versehen hat, oder
6. für den Wahlvorstand erkennbar mehrere oder einen nicht amtlich hergestellten Stimmzettel abgeben oder mit dem Stimmzettel einen weiteren Gegenstand in die Wahlurne werfen will.
[...]

Ein Stimmzettel, der durch die Art, wie er gefaltet ist, durch äußere Markierungen oder ähnliches wiedererkennbar ist, darf nicht in die Wahlurne geworfen werden. Da das Öffnen der versiegelten Wahlurne erst nach Ablauf der Wahlzeit gestattet ist, sollte, sofern sich eine ausreichende Menge Stimmzettel in der Wahlurne befinden, eine Zuordnung zwischen Stimmzettel und Wähler nicht mehr möglich sein.

BWG §39 Ungültige Stimmen, Zurückweisung von Wahlbriefen, Auslegungsregeln
(1) Ungültig sind Stimmen, wenn der Stimmzettel
1. nicht amtlich hergestellt ist,
2. keine Kennzeichnung enthält,
3. für einen anderen Wahlkreis gültig ist,
4. den Willen des Wählers nicht zweifelsfrei erkennen lässt,
5. einen Zusatz oder Vorbehalt enthält.
[...]

Enthält der Stimmzettel Markierungen, könnte er einer Person eindeutig zuzuordnen sein. Daher sind Stimmzettel, die Zusätze enthalten, ungültig. Das ermöglicht es allerdings einem Angreifer, Wähler dazu zu zwingen, ungültig zu stimmen. Mit der Kennzeichnung in Punkt zwei ist das Kreuz gemeint, das den Kandidaten markiert, für den gestimmt werden soll. Da die Markierung, auch wenn ein Stempel dafür verwendet werden würde, immer Spielraum für Abweichungen lässt, kann an dieser Stelle trotzdem nicht vollständig ausgeschlossen werden, dass der Wähler seinen Stimmzettel wiedererkennbar macht.

BWG §32 Unzulässige Wahlpropaganda und Unterschriftensammlung, unzulässige Veröffentlichung von Wählerbefragungen
(1) Während der Wahlzeit sind in und an dem Gebäude, in dem sich der Wahlraum befindet, sowie unmittelbar vor dem Zugang zu dem Gebäude jede Beeinflussung der Wähler durch Wort, Ton, Schrift oder Bild sowie jede Unterschriftensammlung verboten.
(2) Die Veröffentlichung von Ergebnissen von Wählerbefragungen nach der Stimmabgabe über den Inhalt der Wahlentscheidung ist vor Ablauf der Wahlzeit unzulässig.

Auch wenn der Wähler während des Setzens des Kreuzes in der Wahlkabine unbeobachtet ist, soll auch keine anderweitige Beeinflussung möglich sein. Daher sind Hochrechnungen, Erstellen von Teilergebnissen sowie Wahlwerbung in unmitelbarer Nähe zum Wahllokal am Tag der Wahl vor der Schließung der Wahllokale verboten.

1.2.4 Gleich

Eine Wahl muss gleich sein. Jeder Wähler hat eine Stimme und jede Stimme hat das gleiche Gewicht. Eine Gewichtung der Stimmen ist ausgeschlossen, da alle Wähler die gleiche Urne verwenden und die Auszählung öffentlich statt findet. Damit ein Wähler nicht mehrere Stimmzettel abgeben kann, wurden verschiedene Vorkehrungen getroffen.

> **BWO §53 Eröffnung der Wahlhandhlung**
> [...]
> (3) Der Wahlvorstand überzeugt sich vor Beginn der Stimmabgabe davon, dass die Wahlurne leer ist. Der Wahlvorsteher verschließt die Wahlurne. Sie darf bis zum Schluss der Wahlhandlung nicht mehr geöffnet werden.

> **BWO §56 Stimmabgabe**
> [...]
> (6) Der Wahlvorstand hat einen Wähler zurückzuweisen, der
> 1. nicht in das Wählerverzeichnis eingetragen ist und keinen Wahlschein besitzt,
> 2. keinen Wahlschein vorlegt, obwohl sich im Wählerverzeichnis ein Wahlscheinvermerk (§ 30) befindet, es sei denn, es wird festgestellt, dass er nicht im Wahlscheinverzeichnis eingetragen ist,
> 3. bereits einen Stimmabgabevermerk im Wählerverzeichnis hat, es sei denn, er weist nach, dass er noch nicht gewählt hat,
> 4. seinen Stimmzettel außerhalb der Wahlzelle gekennzeichnet oder gefaltet hat oder
> 5. seinen Stimmzettel so gefaltet hat, dass seine Stimmabgabe erkennbar ist, oder ihn mit einem äußerlich sichtbaren, das Wahlgeheimnis offensichtlich gefährdenden Kennzeichen versehen hat, oder
> 6. für den Wahlvorstand erkennbar mehrere oder einen nicht amtlich hergestellten Stimmzettel abgeben oder mit dem Stimmzettel einen weiteren Gegenstand in die Wahlurne werfen will.
> [...]

Diese zwei Abschnitte sollen sicher stellen, dass es keinem Wähler möglich ist mehrere Stimmen abzugeben, oder auch bereits in der Urne befindliche Stimmzettel zu manipulieren. Regelungen zur Briefwahl enthalten weitere Details, wie im Falle mehrerer Stimmzettel pro Wahlberechtigung umzugehen ist.

Kapitel 2

Kryptographische Grundlagen

In diesem Kapitel werden einige Grundlagen sowie Begriffe, Bausteine und kryptographische Primitive vorgestellt, die in kryptographischen Wahlverfahren verwendet werden.

2.1 Informationstheoretische und konditionale Sicherheit

Bei Sicherheitsdefinitionen unterscheidet man häufig die zwei Abstufungen *informationstheoretisch* bzw. *perfekt* (engl. *perfectly* oder *unconditionally*) sicher und *konditional* (im engl. mit *computational* bezeichnet) sicher. Bei konditionaler Sicherheit wird der Angreifer als polynomial beschränkt angenommen, d.h. er kann nur Algorithmen ausführen, deren Laufzeit polynomial in der Länge ihrer Eingabe ist. Deshalb hat sich hier im Englischen der Begriff *computational* etabliert. Die Sicherheit basiert hier immer auf einem Problem, von dem man annimmt, dass es nicht in Polynomialzeit lösbar ist, wie etwa dem DLog-Problem (Abschnitt 2.1.2) oder dem Faktorisierungsproblem (Abschnitt 2.1.3).

Im Gegensatz dazu ist perfekte / informationstheoretische Sicherheit nicht an Voraussetzungen irgendeiner Art geknüpft. Wenn z.B. ein Verschlüsselungsverfahren informationstheoretisch sicher ist, dann kann es selbst ein Angreifer, der unendlich viel Zeit hat und prinzipiell alle Schlüssel durchprobieren könnte, nicht erfolgreich angreifen. Ein Beispiel ist das One Time Pad (siehe Wikipedia): Das Chiffrat enthält ohne den Schlüssel keinerlei *Information* über den Klartext.

2.1.1 Die Polynomialzeit, PPT und die Effizienz

Entsprechend der beiden oben genannten Abstufungen Informationstheoretisch und Konditional sicher ist unsere wichtigste Unterscheidung, was Algorithmenlaufzeit angeht, Polynomialzeit und länger als Polynomialzeit. Sprechen wir von *effizienten* Algorithmen oder effizient lösbaren Problemen, so meinen wir damit stets, dass diese (evtl. probabilistischen) Algorithmen in Polynomialzeit laufen bzw. die Probleme mit einem in Polynomialzeit laufenden Algorithmus lösbar sind (man stelle sich einfach die bekannte Komplexitätsklasse $\mathcal{P}$ vor). Oft spricht man von *PPT-Algorithmen* (probabilistic polynomial time), also probabilistische Algorithmen, die in ihrer Laufzeit durch ein Polynom in der Eingabelänge oder in einem Sicherheitsparameter beschränkt sind.

Genauso kann man zwischen uneingeschränkten und effizienten Angreifern unterscheiden: Uneingeschränkte Angreifer haben beliebig viel Zeit, beliebig viel Rechenleistung und beliebig viel Speicherplatz. Effiziente Angreifer hingegen sind polynomial beschränkt, normalerweise werden sie als PPT-Algorithmen modelliert. Natürlich gibt es noch einige Stufen dazwischen, allerdings mach es meistens keinen Sinn, z.B. subexponentielle Angreifer zu betrachten, wichtig ist meist nur, ob sie effizient sind oder nicht. Oft werden nur effiziente Angreifer betrachtet, was in der realen Welt auch Sinn macht, schließlich hat niemand unendlich viel Rechenleistung geschweige denn unendlich viel Zeit zur Verfügung. Uneingeschränkte Angreifer werden betrachtet, wenn man irgendeine Form von Langzeit-Sicherheit erreichen möchte, z.B. *everlasting privacy* (Niemand soll je das Wahlgeheimnis brechen können, was auch immer passiert), um zu garantieren, dass auch in 50 Jahren oder später niemand die in einer Wahl abgegebene Stimme lernt.

2.1.2 Das DLog-Problem

Sei G eine zyklische Gruppe, $a \in G$. Für ein Element $c = a^x \in G$ ist der Wert x der *diskrete Logarithmus zur Basis* a von c. Das *DLog-Problem* ist nun, zu einem gegebenen Wert $c \in G$ ein x zu finden mit $c = a^x$. In Gruppen $G = \mathbb{Z}_n$ mit ausreichend großem n, in denen die Basis a eine ausreichend große[1] Untergruppe von G erzeugt, so wird das DLog-Problem als schwer ange-

[1] Gruppenordnung min. 2048 bit; Ersetzt man $\mathbb{Z}_n$ durch geeignete elliptische Kurven, ist eine kleinere Gruppenordnung ausreichend.

nommen. Das El-Gamal-Verschlüsselungsverfahren (Abschnitt 2.4.4) und Pedersen-Commitments (Abschnitt 2.2.1) basieren auf dem DLog-Problem. Es sind einige Algorithmen zur Berechnung des diskreten Logarithmus bekannt, die effizienter sind als Brute Force, z.B. der *Baby-Step-Giant-Step*-Algorithmus von Shank (siehe Skript zur Vorlesung Public Key Kryptographie [GBR]), der Laufzeit $O(\sqrt{N})$ hat, wobei N die Ordnung der von der Basis a erzeugten Untergruppe von G ist. Allerdings sind keine DLog-Algorithmen bekannt, die auf klassischen (nicht Quanten-) Rechnern in Polynomialzeit[2] laufen. Auf einem Quantencomputer könnte das DLog-Problem prinzipiell effizient gelöst[3] werden .

2.1.3 Das Faktorisierungsproblem

Das Faktorisierungsproblem ist folgendermaßen definiert: Finde zu einer gegebenen zusammengesetzten Zahl $n \in \mathbb{N}$ einen nicht-trivialen Faktor p. Nicht-trivial heisst dabei $p \notin \{1, n\}$.

Man nimmt an, dass das Faktorisieren von großen (min. 2048 bit) Zahlen auf klassischen (also nicht Quanten-) Rechnern nicht effizient lösbar ist. Es sind bisher keine effizienten Algorithmen bekannt[4], die jede große Zahl faktorisieren können, einen Unmöglichkeitsbeweis gibt es, genau wie beim DLog-Problem auch hier, nicht. Den wird es auch vermutlich so schnell nicht geben, denn ein socher Beweis würde, da Faktorisieren ein NP-Problem ist, $P \neq NP$ implizieren. Das RSA-Verschlüsselungsverfahren basiert zum Beispiel auf dieser Annahme. Das Faktorisierungsproblem wäre jedoch auf einem Quantenrechner effizient mit dem hier nicht behandelten Shor-Algorithmus [Sho97] lösbar.

2.2 Commitments

Ein Commitment-Verfahren (engl. *to commit* = sich festlegen) ermöglicht es, sich auf einen Wert x festzulegen, ohne diesen preiszugeben. Man kann Commitments zu einem späteren Zeitpunkt aufdecken, um zu beweisen, dass es tatsächlich der Wert x war, auf den man sich festgelegt hat.

Ein Commitment-Verfahren besteht entsprechend aus zwei Funktionen *com* und *unveil* und aus zwei Phasen, die diese Funktionen verwenden:

1. In der *Commit-Phase* legt man sich auf einen Wert x fest, indem man $c := com(x)$ berechnet.
2. In der *Unveil-Phase* wird ein Commitment c geöffnet, d.h. man gibt die *unveil-Information* $v := unveil(c)$ bekannt, mit der nachgeprüft werden kann, dass sich mit dem Commitment c tatsächlich auf den Wert x festgelegt wurde.

Commitment-Verfahren müssen die beiden Sicherheitseigenschaften *binding* (“bindend”) und *hiding* (“versteckend”) erfüllen, die im folgenden informell erklärt werden:

- *Bindend (engl. binding)* bedeutet, dass es nicht möglich ist, ein Commitment $c = com(x)$ zu einem anderen Wert als x aufzudecken.
- *Versteckend (engl. hiding)* bedeutet, dass es nicht möglich ist, aus einem Commitment $c := com(x)$ den Wert x zu ermitteln.

Diese beiden Sicherheitseigenschaften können jeweils perfekt (unconditionally) oder konditional (*computationally*) erfüllt sein. Es wurde bewiesen [Dam99], dass ein Commitment nicht gleichzeitig perfekt bindend und versteckend sein kann, d.h. bei einem der beiden Eigenschaften muss man leider auf die informationstheoretische Sicherheit verzichten (Auch die Unmöglichkeit von Quanten-Bit-Commitments wurde bereits bewiesen [MM97]). Für Wahlen werden meistens Commitments verwendet, die nur konditional bindend aber perfekt versteckend sind, um das Wahlgeheimnis auch auf Dauer vorbehaltlos zu schützen. Eine Verletzung der Bindend-Eigenschaft verletzt normalerweise die Verifizierbarkeit. Da verwendete Commitments normalerweise sowieso direkt nach der Wahlphase geöffnet werden, ist davon auszugehen, dass hier konditional bindend ausreicht.

[2] Vorsicht: Die Laufzeit $O(\sqrt{N})$ bei einer exponentiell großen Gruppenordnung N ist natürlich nicht polynomiell!
[3] Shor-Algorithmus [Sho97], in dieser Vorlesung nicht behandelt.
[4] Es gibt einige Algorithmen, die Zahlen schneller als Brute Force faktorisieren können, unter Umständen sogar recht schnell, wenn die zu faktorisierende Zahl bestimmte Eigenschaften besitzt, auf die man dann bei der Parameterwahl achten sollte. Beispielsweise ist die Pollard-$p-1$-Methode (siehe Skript zur Vorlesung Public Key Kryptographie [GBR]) erfolgreich, wenn für einen der Primfaktoren p die Zahl $p-1$ in kleine Faktoren zerfällt.

2.2.1 Beispiel: Pedersen-Commitments

Ein Beispiel für ein Commitment-Verfahren sind Pedersen-Commitments. Diese werden im e-voting-Bereich gerne verwendet, weil sie schöne Eigenschaften besitzen wie z.B. die *Maskierbarkeit* (siehe Abschnitt 2.2.2). Sei G eine zyklische multiplikative Gruppe, in der das DLog-Problem ausreichend schwer lösbar ist (z.B. $G = \mathbb{Z}_p$ für eine ausreichend große Primzahl p) und seien g und h jeweils Erzeuger von G.

Commit-Phase:

1. Wähle eine Zufallszahl r
2. Berechne das Commitment $c := com(x, r) := g^x h^r$.

Unveil-Phase:

3. Decke die Werte x und r auf.

Pedersen-Commitments haben die folgenden Eigenschaften:

Perfekt versteckend: Pedersen-Commitments sind perfekt versteckend, da man sich bei Kenntnis von $log_g(h)$ zu jedem beliebigen Commitment $c = g^x h^r$ zu jedem beliebigen Wert $y \in G$ eine Zahl r' ausrechnen könnte, so dass $c = g^y h^{r'}$. (Wie, das bleibt dem Leser dieses Skripts erstmal als Übung überlassen.) Genau genommen ist c also gleichzeitig ein Commitment auf alle Elemente in G, deshalb kann selbst durch durchprobieren nicht heraus gefunden werden, für welchen Wert das Commitment erstellt wurde. Es könnte prinzipiell jeder sein.

Konditional bindend: Der gleiche Grund, der den Pedersen-Commitments die perfekte hiding-Eigenschaft gibt, ist allerdings auch dafür verantwortlich, dass sie nur konditional bindend (in diesem Fall basierend auf dem DLog-Problem) sind: In jedem Commitment kann prinzipiell jeder Wert aus der Klartextmenge stecken. Mit Kenntnis von $log_g(h)$ ist es leicht, zu einem Pedersen-Commitment c und jedem beliebigen Wert $y \in G$ eine Zahl r' auszurechnen, für die $c = g^y h^{r'}$ gilt. Damit könnte man mit Kentniss von $log_g(h)$ das Commitment zu jedem beliebigen Wert öffnen. Damit wäre die binding-Eigenschaft verletzt. Ohne Kenntnis von $log_g(h)$ ist es jedoch ausreichend schwer zu gegebenem c und y ein r' zu finden für das $c = g^y h^{r'}$ gilt. Natürlich sollte man bei Pedersen-Commitments immer darauf achten, dass derjenige, der die Commitments erstellt, nicht die Parameter g und h frei wählen darf, sonst könnte er h so wählen, dass er $log_g(h)$ kennt, indem er $h = g^k$ für ein beliebiges k berechnet.

2.2.2 Maskierbarkeit von Pedersen-Commitments

Pedersen-Commitments werden in kryptographischen Wahlverfahren nicht nur wegen ihrer perfekten hiding-Eigenschaft verwendet, sondern auch wegen einer weiteren sehr nützlichen Eigenschaft: Sie sind *maskierbar*. Das bedeutet, man kann aus einem Pedersen-Commitment c auf einen Wert m ein neues Commitment c' auf den gleichen Wert m erstellen, indem man den Zufallswert ändert, mit dem der Parameter h potenziert wurde: Gegeben sei das Pedersen-Commitment $c = g^m h^r$ auf die Nachricht m mit Zufall r. Zur Maskierung wählt man eine weitere Zufallszahl s und berechnet

$$c' = c \cdot h^s.$$

Damit gilt

$$c' = g^m h^r h^s = g^m h^{r+s},$$

c' ist also auch ein Commitment auf m mit neuem Zufall $r + s$.

Pedersen-Commitments können dem zu Folge ohne Kenntnis des Klartextes maskiert werden. Außerdem ist den Commitments c und c' wegen der perfekten hiding-Eigenschaft nicht anzusehen, dass sie Commitments auf den gleichen Wert sind. Der Ersteller der Maskierung kann diese Tatsache aber beweisen, ohne m zu kennen oder gar offenzulegen. Er muss dazu nur den Maskierungszufall s offenlegen. Damit kann nachgeprüft werden, dass $c' = c \cdot h^s$. (Es sei hier zu bemerken, dass dieser Beweis der Maskierung von c nur überzeugend ist, wenn der Ersteller der Maskierung $log_g(h)$ nicht kennt, und nicht effizient diskrete Logarithmen finden kann. Dem Leser wird empfohlen, nachzurechnen, warum.) Die ohne Kenntnis von m durchführ- und beweisbare Maskierbarkeit ist dann nützlich, wenn man mehrere Commitments mittels eines verifizierbaren *Shuffle* (siehe Abschnitt

2.5) öffnen will. In dem Fall hat man eine Menge an Commitments, die man, ggf. verteilt auf mehrere Server, so öffnen möchte, dass dabei niemand die Zuordnung zwischen den Commitments und den beinhalteten Nachrichten lernt. Anwendung findet die Maskierung von Commitments u.a. im Wahlverfahren Bingo Voting (Abschnitt 5.1) sowie dem Verfahren von Moran und Naor (Abschnitt 3.2.1).

Maskierungen von Commitments können als Analog zur *Re-Encryption* (Wiederverschlüsselung, Abschnitt 2.4.3) von Chiffraten angesehen werden: Die beinhaltete Nachricht bleibt die selbe, während der Zufall verändert wird.

2.3 Zero-Knowledge-Beweise

Zero-Knowledge-Beweise werden hier nur informell beschrieben. Bei Zero-Knowledge-Beweisen (kurz: ZK-Beweise) handelt es sich um meist interaktive Beweissysteme, mit denen jemand beweisen kann, dass ihm ein Geheimnis bekannt ist, ohne dieses preiszugeben.

2.3.1 Informelle Definition

Es soll in den meisten Fällen die Gültigkeit einer Aussage A gezeigt werden (z.B. die Kenntnis eines Geheimnisses). Ein ZK-Beweis besteht aus einer Interaktion zwischen dem *Beweiser* (engl. *Prover*) P, der eine Aussage A beweisen möchte, und einem *Verifizierer* V, den es von der Wahrheit der Aussage A zu überzeugen gilt. Der Beweiser verfügt im Allgemeinen über einen Zeugen w (von engl. *witness*, z.B. das Geheimnis), aus dessen Existenz die Aussage folgt, der jedoch geheim gehalten werden soll.

Ein ZK-Beweis muss die folgenden drei Eigenschaften erfüllen:

Completeness (Vollständigkeit) Ist die Aussage A wahr, so akzeptiert der Verifizierer V den Beweis von P (mit überwältigender Wahrscheinlichkeit).

Soundness (Zuverlässigkeit) Ist die Aussage A falsch, also der Beweiser P unehrlich, so lehnt der Verifizierer V den Beweis (mit überwältigender Wahrscheinlichkeit) ab. Da ein ZK-Beweis normalerweise aus mehreren Runden besteht, reicht es, wenn in jeder einzelnen Runde der Beweis mit einigermaßen hoher Wahrscheinlichkeit abgelehnt wird, z.B. 50%. Um insgesamt den Verifizierer zu überzeugen, müsste der Beweiser den Verifizierer auch in jeder einzelnen Runde überzeugen. Mit höherer Anzahl Runden kann man die Soundness also soweit steigern, dass der Beweis eines unehrlichen Beweisers insgesamt mit überwältigender Wahrscheinlichkeit abgelehnt wird.

Simulatability (Simulierbarkeit, Zero-Knowledge-Eigenschaft) Der Verifizierer lernt aus dem Beweis nicht mehr als die Tatsache, dass A wahr ist. Ein Dritter lernt aus dem Beweis nichts. V kann also den Beweis nicht weiterverwenden und damit einen Dritten von der Aussage A überzeugen. Der Beweiser P kann also V beweisen, dass er ein Geheimnis w kennt, ohne dass V danach einem Dritten beweisen könnte, dass er das Geheimnis selbst kennt. Modelliert wird diese sogenannte Zero-Knowledge-Eigenschaft durch die *Simulierbarkeit*: Das Protokolltranskript eines ZK-Beweises ist so geartet, dass sich der Verifizierer auch selbst ein Transkript ausdenken könnte, das für Dritte ununterscheidbar von einem echten Beweisdurchlauf ist. Der Verifizierer kann den Beweis also *simulieren*.

Die Simulierbarkeit kommt normalerweise dadurch zustande, dass der Verifizierer dem Beweiser eine *Challenge* schickt, auf die der Beweiser reagieren muss. Diese Challenge besteht normalerweise aus Zufallsbits. Ein Beispiel ist in Abschnitt 2.3.2 zu sehen. Hier kann sich ein Verifizierer einen von einem echten Protokollablauf ununterscheidbaren Ablauf ausdenken (den Beweis simulieren), da er sich die Challenge selbst stellt und dementsprechend lösen kann ohne das Geheimnis zu kennen.

ZK-Beweise in dieser Form können nur interaktiv stattfinden. Für die Zwecke von elektronischen Wahlen verwendet man hier die sogenannte *Fiat-Shamir-Heuristik* (siehe Abschnitt 2.3.3), um ZK-Beweise nicht-interaktiv zu machen.

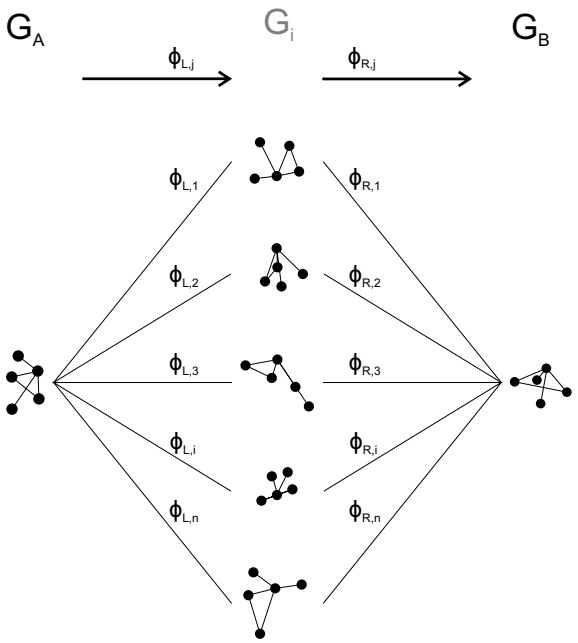

Abbildung 2.1: Schemadarstellung des Graph-Isomorphie-Zero-Knowledge-Beweises

2.3.2 Beispiel: Graphisomorphie

Der Beweis, dass zwei Graphen isomorph zueinander sind, ist eines der Standardbeispiele für Zero-Knowledge-Beweise. Der Beweiser P veröffentlicht zwei Darstellungen G_A und G_B eines Graphen, behält aber den Isomorphismus ϕ zwischen den beiden Darstellungen für sich. Damit ist die Aussage, die P gegenüber dem Verifizierer V beweisen möchte, die folgende: "Ich kenne einen Isomorphismus, der G_A auf G_B abbildet." Der Zeuge, den P zur Verfügung hat, aber nicht preis geben möchte, ist der Isomorphismus ϕ.

Der Beweiser überzeugt den Verifizierer in mehreren Runden, schematisch dargestellt in Abbildung 2.1, dass er ϕ kennt. Dazu erzeugt er eine zufällige Abbildung $\phi_{L,i}$ und wählt $\phi_{R,i}$ so, dass $\phi_{L,i} \circ \phi_{R,i} = \phi$. Der Beweiser wendet $\phi_{L,i}$ auf den Graphen G_A an, und schickt den daraus resultierenden Graphen G_i an den Verifizierer. Dieser kann sich nun entweder $\phi_{L,i}$ oder $\phi_{R,i}$ zeigen lassen und entsprechend prüfen, ob der Zusammenhang zwischen G_A und G_i oder der Zusammenhang G_i und G_B korrekt ist. Könnte er beide Zusammenhänge prüfen, so wäre er in der Lage ϕ zu berechnen. Genauso muss P aber auch ϕ kennen, wenn er in der Lage ist beide Antworten zu geben. Solange er die Wahl des Verifizierers nicht vorhersehen kann, würde er mit einer Wahrscheinlichkeit von $\frac{1}{2}$ auffallen, wenn er ϕ nicht kennt. In diesem Fall wäre er nicht in der Lage aus $\phi_{L,i}$ $\phi_{R,i}$ zu berechnen. Er könnte allenfalls G_i aus G_B generieren (indem er $\phi_{R,i}$ wählt), könnte aber dann $\phi_{L,i}$ nicht angeben.

Die Wahrscheinlichkeit, dass dem Verifizierer nicht auffällt, wenn der Beweiser die Abbildung ϕ nicht kennt, sinkt von Runde zu Runde um Faktor 1/2.

2.3.3 Fiat-Shamir-Heuristik

Zero-Knowledge-Beweise wie oben beschrieben sind in der Regel interaktive Beweise: Der Verifizierer schickt dem Beweiser eine für ihn unvorhersagbare *Challenge*, auf die der Beweiser antworten muss. Fiat et. al. haben in [FS86] vorgeschlagen, stattdessen als Challenge ein sogenanntes *Random Oracle* zu benutzen. Ein *Random Oracle* kann als Funktion $\mathcal{O}$ gesehen werden, die eine Eingabe x auf echten Zufall abbildet. Für jede neue Eingabe gibt das Oracle eine neue, von vorhergehenden Ein- und Ausgaben unabhängige Zufallszahl aus. Für eine bereits getätigte Eingabe wird derselbe Zufall zurückgegeben, der bei der ersten Eingabe gewählt wurde. Random Oracles sind leider nur Gedankenkonstrukte. Es wurde in [CGH04] bewiesen, dass eine reale Implementierung eines Random Oracles nicht existieren kann. Oft ist aber eine geeignete kryptographische Hashfunktion über ausreichend unvorhersehbare Daten als Ersatz gut genug. Beispiele hierfür werden in verschiedenen, im Skript beschriebenen Korrektheitsbeweisen von Wahlprotokollen noch zu finden sein.

2.4 Verschlüsselungsverfahren

Verschlüsselungsverfahren spielen bei elektronischen Wahlen eine große Rolle: Um das Wahlgeheimnis zu wahren, sollte eine Wählerstimme, oder Daten, aus denen Information über die Wählerstimme hervorgehen könnte, natürlich nicht unverschlüsselt übertragen werden. Bei Wahlen haben wir oft spezielle Anforderungen an Verschlüsselungsverfahren wie die Wiederverschlüsselbarkeit bei Re-Encryption-Mixnetzen oder die Homomorphie bei einer homomorphen Stimmauszählung. In diesem Abschnitt werden Beispiele für solche Verschlüsselungsverfahren vorgestellt.

2.4.1 Symmetrische und Asymmetrische Verschlüsselung

Wir unterscheiden zwischen symmetrischen und asymmetrischen Verschlüsselungsverfahren.

Symmetrische Verfahren Bei symmetrischen Verfahren ist der Schlüssel zum Verschlüsseln gleich dem Schlüssel zum Entschlüsseln, oder die beiden Schlüssel sind sehr leicht auseinander ableitbar. Symmetrische Verfahren sind i.d.R. deutlich effizienter als asymmetrische Verfahren. Bekannte Beispiele für symmetrische Verfahren sind der DES[5] (*Data Encryption Standard*) und der AES[6] (*Advanced Encryption Standard*).

Asymmetrische Verfahren Bei asymmetrischen Verfahren werden zum Ver- und Entschlüsseln unterschiedliche Schlüssel verwendet. Dabei darf der Verschlüsselungsschlüssel (*öffentlicher Schlüssel*, meist *pk* wie *public key*) öffentlich bekannt sein. Jeder kann ihn benutzen, um Nachrichten zu verschlüsseln. Der zugehörige Entschlüsselungsschlüssel (*geheimer Schlüssel*, meist *sk* für *secret key* genannt) muss geheim bleiben. Nur der berechtigte Empfänger einer Nachricht kann mit dem nur ihm bekannten geheimen Entschlüsselungsschlüssel wieder entschlüsseln. Der geheime Schlüssel darf dabei ohne (geheime!) Zusatzinformation nicht effizient aus dem öffentlichen Schlüssel berechenbar sein. Als Beispiel für ein asymmetrisches Verfahren wird hier das ElGamal-Verschlüsselungsverfahren vorgestellt (siehe Anschnitt 2.4.4).

2.4.2 Deterministische und nicht-deterministische Verschlüsselung

Neben der Unterscheidung symmetrisch und asymmetrisch gibt es außerdem die Unterscheidung zwischen deterministischen und nicht-deterministischen Verschlüsselungsverfahren. Deterministische Verfahren arbeiten ganz ohne Zufall. Das bedeutet, dass der gleiche Klartext immer auf das gleiche Chiffrat abgebildet wird. Bei einer kleinen Klartextmenge, wie z.B. die Menge möglicher Wählerstimmen, kann bei einem deterministischen asymmetrischen Verfahren sehr schnell den zum Chiffrat gehörigen Klartext heraus gefunden werden, indem jeder mögliche Klartext verschlüsselt und mit dem vorliegenden Chiffrat verglichen wird. Aus diesem Grund werden bei elektronischen Wahlen für die meisten (wenn nicht alle) Anwendungen nicht-deterministische Verschlüsselungsverfahren verwendet. Hier geht in die Berechnung des Chiffrats ein Zufallswert mit ein, so dass bei der Verschlüsselung der gleichen Nachricht praktisch immer ein anderes Chiffrat herauskommt. So lässt sich durch durchprobieren der Klartext nicht erraten. Es lässt sich zwei Chiffraten auch nicht ansehen, ob sie den gleichen Klartext enthalten. Das in Abschnitt 2.4.4 vorgestellte ElGamal-Verfahren ist nicht-deterministisch. Ein Beispiel für ein deterministisches Verfahren wäre RSA in seiner Urversion (siehe z.B. [GBR]).

2.4.3 Wiederverschlüsselung

Die Möglichkeit der Wiederverschlüsselung (engl. *Re-Encryption*) ist eine nützliche Eigenschaft mancher nicht-deterministischer Verschlüsselungsverfahren, die es erlaubt, aus einem gegebenen Chiffrat ein neues Chiffrat zu erzeugen, das den selben Klartext enthält. Dies geschieht durch Neurandomisierung des Chiffrats, d.h. man verändert den Zufall, der in die Berechnung des Chiffrats eingegangen ist, lässt aber den Klartext gleich. Diese Operation kann oft direkt auf dem Chiffrat durchgeführt werden ohne die Nachricht oder den geheimen Schlüssel zu kennen. Ein Beispiel hierfür wird in Abschnitt 2.4.4.1 bei der ElGamal-Wiederverschlüsselung gezeigt.

Wiederverschlüsselung kann als Analog zur Maskierung von Commitments gesehen werden (siehe Abschnitt 2.2.2): auch hier bleibt die Nachricht gleich, aber der verwendete Zufall wird verändert.

[5] http://de.wikipedia.org/wiki/Data_Encryption_Standard
[6] http://de.wikipedia.org/wiki/Advanced_Encryption_Standard

Anwendung findet die Wiederverschlüsselung z.B. in *Re-Encryption-Mixnetzen* (Abschnitt **??**) oder bei Internetwahlverfahren wie z.B. Helios, um zu verschleiern, welches Chiffrat von welchem Wähler stammt.

2.4.4 El-Gamal-Verschlüsselung

Das ElGamal-Verschlüsselungsverfahren [ElG85] wurde von Taher El Gamal entworfen und ist nach RSA eins der ersten bekannten asymmetrischen Verschlüsselungsverfahren. Das ElGamal-Verfahren hat gegenüber RSA den Vorteil, dass es nicht-deterministisch ist, d.h. gleiche Nachrichten werden auf unterscheidliche Chiffrate abgebildet. Das ElGamal-Verfahren basiert auf der DLog-Annahme (Abschnitt 2.1.2): Solange es schwer ist, in der verwendeten Gruppe den diskreten Logarithmus zu berechnen, ist der geheime Schlüssel nicht effizient aus dem öffentlichen Schlüssel berechenbar.

Schlüsselerzeugung Sei G eine multiplikative zyklische Gruppe mit Primzahlordnung p, z.B. die Einheitengruppe $\mathbb{Z}_p^*$ von $\mathbb{Z}_p$. Für die Primzahl p sollte gelten, dass $p-1$ einen großen Primfaktor besitzt, z.B. $p = 2q+1$ für eine Primzahl q. Gilt dies nicht, so sind Algorithmen, die das Verfahren brechen, mit höherer Wahrscheinlichkeit oder schneller erfolgreich. Mehr dazu erfährt man in unserer Vorlesung *Asymmetrische Verschlüsselungsverfahren*[7]. Sei $g \in G$ ein Erzeuger der Gruppe G. Das bedeutet, dass $< g >= \{g^0, g^1, g^2, \ldots, g^{p-1}\} = G$. Als geheimen Schlüssel wählt man ein zufälliges $x \in \mathbb{N}$. Der zugehörige öffentliche Schlüssel ist $y := g^x$. Die Gruppe G, der Erzeuger g und der öffentliche Schlüssel y werden öffentlich bekannt gegeben. Der geheime Schlüssel x wird selbverständlich geheim gehalten.

Verschlüsseln Um eine Nachricht m mit dem öffentlichen Schlüssel y zu verschlüsseln, wählt man zunächst eine Zufallszahl r ($r \in \mathbb{Z}_p$ falls $G = \mathbb{Z}_p^*$). Danach berechnet man die beiden Werte $A := g^r$ und $B := y^r m$. Das Chiffrat c ist dann das Tupel

$$c := (A, B) = (g^r, y^r m).$$

Die Berechnungen finden natürlich in der Gruppe G statt, d.h. für $G = \mathbb{Z}_p^*$ ist $A = g^r \mod p$ und $B = y^r m \mod p$.

Entschlüsseln Um ein Chiffrat $c = (A, B)$ zu entschlüsseln, benötigt man den geheimen Schlüssel x, jedoch nicht den zum Verschlüsseln verwendeten Zufall r. Man berechnet einfach

$$m = A^{-x} B.$$

Es gilt

$$A^{-x} B = (g^r)^{-x} (g^x)^r m = g^{-rx} g^{xr} m = m.$$

2.4.4.1 Homomorphie und Wiederverschlüsselbarkeit

Das ElGamal-Verschlüsselungsverfahren ist multiplikativ homomorph (bzgl. komponentenweiser Multiplikation): Gegeben zwei Chiffrate $c_1 = (g^{r_1}, y^{r_1} m_1)$, $c_2 = (g^{r_2}, y^{r_2} m_2)$, gilt für das Produkt c der Chiffrate:

$$c := c_1 c_2 = (g^{r_1} g^{r_2}, y^{r_1} m_1 y^{r_2} m_2) = (g^{r_1 + r_2}, y^{r_1 + r_2} m_1 m_2),$$

c ist also eine Verschlüsselung des Produkts $m_1 m_2$ der beiden verschlüsselten Klartexte, mit Zufall $r_1 + r_2$.

Wiederverschlüsselung Die Homomorphie des Verfahrens kann ausgenutzt werden, um Wiederverschlüsselung zu realisieren: Dazu wird das Chiffrat der 1 mit dem gegebenen Chiffrat multipliziert. Dadurch wird nur der Zufall der Verschlüsselung, aber nicht die Nachricht verändert. Dazu ist weder die Kenntnis über den Zufall des ursprünglichen Chiffrats noch die der Nachricht notwendig. Es wird lediglich der öffentliche Schlüssel y benötigt.

[7] Auf der Webseite zur Vorlesung http://www.iks.kit.edu/index.php?id=asv-ws12 steht ein Skript zur Vorgängervorlesung *Public Key Kryptographie* zum Download bereit, in dem entsprechende Algorithmen zur Berechnung des diskreten Logarithmus vorgestellt werden, die effizienter werden, je "ungeschickter" die Gruppe G gewählt wird.

Formaler: Sei $c = (g^r, y^r m)$ das Chiffrat, auf dem die Wiederverschlüsselung durchgeführt werden soll. Wähle eine weitere Zufallszahl r', berechne die Verschlüsselung der 1:

$$c_1 = (g^{r'}, y^{r'}),$$

und daraus dann das wiederverschlüsselte Chiffrat

$$c' = (g^r, y^r m)(g^{r'}, y^{r'}) = (g^{r+r'}, y^{r+r'} m).$$

Damit ist c' eine Verschlüsselung der Nachtricht m mit neuem Zufall $r + r'$.

2.4.4.2 Beweis für korrektes Entschlüsseln

Eine Möglichkeit, die korrekte Entschlüsselung zu beweisen, ist natürlich immer, den Verschlüsselungszufall offenzulegen, dann kann man zum Verifizieren mit genau diesem Zufall verschlüsseln und prüfen, dass das richtige Chiffrat rauskommt. Manchmal ist dies jedoch nicht die brauchbarste Methode, z.B., wenn der Zufall der entschlüsselnden Partei garnicht bekannt ist. Für das ElGamal-Verfahren gibt es für diesen Fall mit dem Chaum-Pedersen-Protokoll[8] [CP93] eine Alternative, die z.B. im Helios-Verfahren (Abschnitt 6.1) verwendet und im entsprechenden Papier [Adi] nochmals vorgestellt wird:

Chaum-Pedersen-Protokoll Gegeben seien ein Chiffrat (α, β) und ein Klartext m. Sei x der geheime Schlüssel und $y = g^x$ der öffentliche Schlüssel. Es soll bewiesen werden, dass $(\alpha, \beta) = (g^r, (g^x)^r m)$ für ein r, dass (α, β) also eine Verschlüsselung von m ist. Das Chaum-Pedersen-Protokoll beweist die Logarithmengleichheit

$$\log_g y = \log_\alpha \frac{\beta}{m},$$

die genau dann gilt, wenn (α, β) eine Verschlüsselung von m ist, denn genau dann gilt:

$$\log_g y = \log_g g^x = x = \log_{g^r} (g^r)^x = \log_{g^r} \frac{(g^r)^x m}{m} = \log_\alpha \frac{\beta}{m}.$$

Der Beweis wird geführt durch ein interaktives Protokoll zwischen einem Beweiser P, der das korrekte Entschlüsseln beweisen möchte, und einem Verifizierer V, der dies überprüfen möchte. Es hat folgende Schritte:

1. P wählt ein zufälliges $w \in \mathbb{Z}_q$ und schickt $a := g^w$ sowie $b := \alpha^w$ an V.
2. V schickt eine zufällige Challenge $c \in \mathbb{Z}_q$ an P.
3. P schickt $t = w + xc$ an V.
4. V prüft, ob $g^t = ay^c$ und $\alpha^t = b(\frac{\beta}{m})^c$. Dabei wird mit der ersten Gleichung geprüft, ob für die Berechnung von t die richtige Challenge verwendet wurde, und die zweite Gleichung kann nur gelten, wenn (α, β) eine Verschlüsselung von m ist. Man sieht dies leicht, wenn man die Gleichungen ausführlich hinschreibt (z.B. g^r statt α u.s.w.), was zum besseren Verständnis hier empfohlen wird.

2.4.4.3 Beweis Klartextkenntnis

Ein dem im vorigen Abschnitt vorgestellten Chaum-Pedersen-Protokoll sehr ähnliches Protokoll ist das *Schnorr-Protokoll*. Mit diesem Protokoll kann man unter Anderem beweisen, dass man zu einem bestimmten ElGamal-Chiffrat den Klartext kennt. Dies beweist man hier indirekt, indem man die Kenntnis des Verschlüsselungszufalls beweist. Wer den kennt, kann den Klartext nämlich prinzipiell ohne Kenntnis des geheimen Schlüssels aus dem Chiffrat berechnen. Wie das geht, bleibt dem Leser hier als kleine Fingerübung überlassen.

[8] Das Chaum-Pedersen-Protokoll ist ursprünglich ein Signaturverfahren, der Algorithmus zum Prüfen der Signatur kann jedoch hier angewendet werden, um die korrekte Entschlüsselung zu prüfen.

Schnorr-Protokoll Gegeben sei ein $y = g^r$ mit dem Verifizierer unbekanntem r, dabei sei g ein Erzeuger einer zyklischen Gruppe, in der das DLog-Problem schwer ist. Es gilt die Kenntnis eines r mit $y = g^r$ zu beweisen. Das Protokoll ist ein interaktiver Beweis, der folgendermaßen abläuft:

1. Der Beweiser wählt ein zufälliges a, berechnet $t := g^a$ und schickt t dem Verifizierer.
2. Der Verifizierer wählt eine zufällige Challenge c und schickt sie dem Beweiser.
3. Der Beweiser berechnet $s := a + cr$ und schickt s dem Verifizierer.
4. Der Verifizierer prüft, ob $g^s = ty^c$

2.4.4.4 Exponentielles El Gamal

ElGamal ist multiplikativ homomorph. Um eine homomorphe Stimmauszählung zu ermöglichen, wird jedoch normalerweise ein additiv homomorphes Verfahren benötigt. Das ElGamal-Verfahren lässt sich zu diesem Zweck relativ einfach in ein additiv homomorphes Verfahren umwandeln: Sei G wieder eine zyklische Gruppe mit Primzahlordnung $|G| = p$, $g \in G$ ein Erzeuger von G, x der geheime und $y = g^x$ der öffentliche Schlüssel. Verschlüsselt wird nun, indem mit dem originalen ElGamal-Verfahren nicht direkt m verschlüsselt wird, sondern g^m. D.h. das Chiffrat c berechnet sich nun als

$$c := (A, B) := (g^r, y^r g^m).$$

Das Entschlüsseln wird nun etwas schwieriger, denn es muss zur Berechnung von m der diskrete Logarithmus in G gezogen werden: Für ein gegebenes Chiffrat $c = (A, B)$ ist der zugehörige Klartext

$$m = \log_g (A^{-x} B).$$

Dies ist für eine kleine Klartextmenge, wie die Menge der möglichen Wahlergebnisse eine ist, mit vertretbarem Aufwand machbar (Brute Force, Baby-Step-Giant-Step,...).

2.4.4.5 Threshold El Gamal

Threshold-ElGamal wird hier nicht ausführlich behandelt. Threshold-Verfahren sind jedoch eine wichtige Primitive für kryptographische Wahlverfahren, deshalb soll hier zumindest die Idee vermittelt werden. Für eine ausführlichere Beschreibung siehe [Ped91] und darin aufgeführte Quellen.

Um zu verhindern, dass eine Instanz alleine einzelne Stimmzettel entschlüsseln und damit das Wahlgeheimnis brechen kann, werden oft sogenannte k-aus-t-Threshold-Verfahren (Schwellwertverfahren) verwendet: Der geheime Schlüssel zum Entschlüsseln wird aufgeteilt auf t Parteien, so dass mindestens k dieser t Teilnehmer zusammenarbeiten müssen, um gemeinsam entschlüsseln zu können. Schließen sich beliebige k aus diesen t Teilnehmern zusammen, so können sie gemeinsam entschlüsseln. Weniger als k Teilnehmer haben jedoch zusammen keinerlei Information über den geheimen Schlüssel und können nicht entschlüsseln. Dabei ist wünschenswert, dass bei der Entschlüsselung der geheime Schlüssel selbst nicht rekonstruiert wird, damit auch nach einem Entschlüsselungsvorgang weiterhin nur gemeinsam entschlüsselt werden kann. Aus dem ElGamal-Verschlüsselungsverfahren kann man ein solches Verfahren konstruieren:

Schlüsselerzeugung Seien im Folgenden $A_1, \ldots, A_t$ die Parteien, unter denen der geheime Schlüssel aufgeteilt werden soll. Wir beschreiben zuerst die Idee für $k = t$: Bei der Schlüsselerzeugung wird ein Schlüsselpaar (pk, sk) gemeinsam von den t Parteien erzeugt, die später zusammen entschlüsseln können sollen, und zwar auf eine Weise, dass jeder Teilnehmer A_i sein Schlüssel-Share s_i bekommt, aber niemand Information über den kompletten Schlüssel sk hat. Seien wieder p und q Primzahlen, q teilt $p-1$, G sei die eindeutige Untergruppe von $\mathbb{Z}_p^*$ mit Ordnung q. Sei g ein Erzeuger der Gruppe G. Weiterhin sei C ein Commitment-Verfahren (z.B. ein Pedersen-Commitment, siehe Abschnitt 2.2.1), $C(m)$ notiert dabei ein Commitment auf eine Nachricht m. Die Schlüsselerzeugung (aus [Ped91] und [CGS97]) funktioniert nun allgemein so:

1. Jede Partei A_i wählt ein $s_i \in \mathbb{Z}_q$ zufällig und gleichverteilt und berechnet $pk_i = g^{s_i}$. Danach committet sich A_i mit $c_i := C(pk_i)$ auf ihr Share s_i. Das Commitment c_i wird an alle anderen Parteien verteilt.
2. Sind die Commitments aller Parteien verteilt, öffnet jede Partei A_i ihr Commitment c_i.

3. Der öffentliche Schlüssel pk wird berechnet:

$$pk = \Pi_{i=1}^{n} pk_i.$$

Um daraus ein k-aus-t-Schema zu machen (aus t Teilnehmern kann eine beliebige Teilmenge von k Teilnehmern rekonstruieren), wird das obige Vorgehen so erweitert, dass die s_i derart zusammenhängen, dass sk aus einer beliebigen Untermenge von k aus den t Punten durch Lagrange-Interpolation rekonstruiert werden kann:

Sei M die Menge der Indizes der k zur Rekonstruktion verwendeten Punkte. Dann soll gelten:

$$sk = \sum_{j \in M} s_j \lambda_j,$$

wobei

$$\lambda_j = \prod_{l \in M} \frac{s_l}{s_l - s_j}.$$

Für das genaue Vorgehen siehe z.B. [Ped91].

Verschlüsseln Verschlüsselt wird wie beim normalen ElGamal-Verfahren mit Public Key $pk = g^{sk}$.

Entschlüsseln Sei $c = (A, B)$ das zu entschlüsselnde Chiffrat, $A_1, \ldots, A_k$ die k entschlüsselnden Teilnehmer. Sie entschlüsseln c folgendermaßen gemeinsam, ohne den geheimen Schlüssel zu rekonstruieren:

1. Jede Partei A_j veröffentlicht $w_j := A^{s_j}$ und beweist zero-knowledge, dass

$$log_g(pk_j) = log_A(w_j),$$

ähnlich dem Beweis für korrektes Entschlüsseln in Abschnitt 2.4.4.2.

2. Entschlüsseln:

$$m = \frac{B}{\prod_{j=1}^{k} w_j^{\lambda_j}}$$

Wer genau hinschaut, sieht, dass das Geheimnis nur im Exponenten rekonstruiert wird, so kann entschlüsselt werden, ohne dass eine der Parteien Information über den Schlüssel lernt. Das Geheimbleiben des Schlüssels basiert natürlich dadurch, wie die ElGamal-Verschlüsselung selbst, auf dem DLog-Problem (Siehe Abschnitt 2.1.2).

Wer noch genauer hingeschaut hat, dem ist auch aufgefallen, dass die k Teilnehmer durchaus in der Lage wären, das Geheimnis sk gemeinsam zu rekonstruieren. Wir können jedoch davon ausgehen, dass die k Teilnehmer als Gruppe vertrauenswürdig sind und das nicht tun: Weigert sich auch nur eine der k Parteien (bzw. $t - k + 1$ der t Parteien), an der Schlüsselrekonstruktion teilzunehmen, so kann die Rekonstruktion - genau wie das Entschlüsseln - nicht stattfinden. So können wir im konkreten Fall von Wahlverfahren annehmen, dass die t Teilnehmer als Gruppe sich weigern, einzelne Wählerstimmen zu entschlüsseln, solange eine Verbindung zwischen Wähler und Chiffrat der Wählerstimme besteht.

2.5 Verifizierbares Mischen (engl: *verifiable Shuffle*)

Verifizierbares Mischen kann man sich vorstellen als das kryptographische Analog zur Wahlurne, in die verifizierbar eine Menge an Stimmen hineingeht, zufällig gemischt wird, und in anderer Reihenfolge wieder herauskommt. Wird das Mischen von mehreren Servern ausgeführt, so spricht man von einem *Mixnetz* (engl. *mixnet*), dessen beteiligte Server heissen *Mixserver*. Den Vorgang des Mischens nennen wir einen *Mix* (engl. *mix* oder *shuffle*; oft wird mit *shuffle* auch das Ergebnis eines Mixes bezeichnet, also die permutierten maskierten Chiffrate).

Ganz allgemein bekommt ein Mix eine Menge an Chiffraten oder Commitments als Eingabe, und gibt diese wiederverschlüsselt bzw. maskiert in permutierter Reihenfolge wieder aus, so dass nicht erkennbar ist, welches Ausgabechiffrat/Commitment welchen Eingabechiffrat/Commitment entspricht. Nun gilt es zu beweisen, dass der Mix korrekt durchgeführt wurde, d.h. dass keine Chiffrate gegen völlig andere ausgetauscht wurden, oder Chiffrate weggenommen und dafür andere mehrmals verwendet wurden. Dafür gibt es verschiedene Beweistechniken.

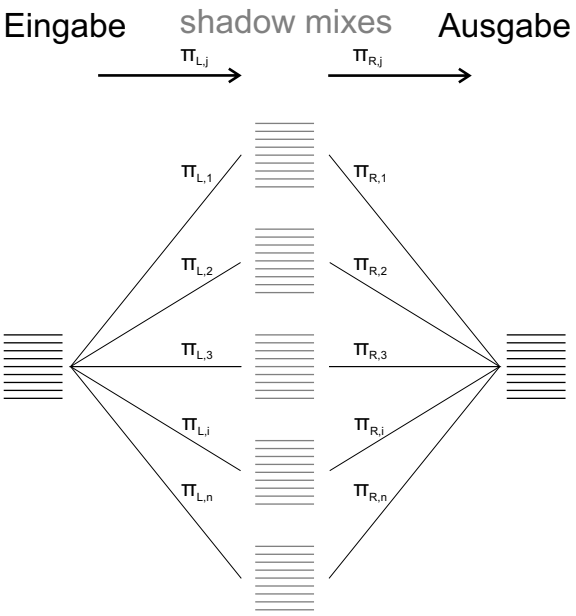

Abbildung 2.2: Zero-Knowledge-Beweis für korrektes Mischen mit Shadow Mixes. Vorsicht Fehler im Bild: Es sollten k Shadow Mixes sein, nicht n, wird demnächst verbessert.

2.5.1 Reencryption-Mix mit Shadow Mixes

Eine Möglichkeit, das korrekte Mischen von Chiffraten zu beweisen, ist ein Zero-Knowledge-Beweis, der bei genauerer Betrachtung sehr dem Beweis der Graphisomorphie (siehe Abschnitt 2.3.2) ähnelt. Abbildung 2.2 stellt den Beweis schemenhaft dar. Wir erklären den Beweis nun für einen Shuffle (Permutation und Wiederverschlüsselung) von Chiffraten. Für Commitments ersetze man einfach Chiffrat durch Commitment und Wiederverschlüsselung durch Maskierung und gehe genauso vor. Ein Mixserver möchte beweisen, dass eine Liste von n Chiffraten $\mathcal{C}' = \{c'_1, \ldots, c'_n\}$ ein *shuffle* einer Liste $\mathcal{C} = \{c_1, \ldots, c_n\}$ ist, d.h. $\mathcal{C}'$ ist aus $\mathcal{C}$ durch Permutation und Wiederverschlüsselung der $c_1, \ldots, c_n$ entstanden. Genauer gibt es für jedes $c_i \in \mathcal{C}$ genau ein $c'_j \in \mathcal{C}'$ mit $j = \pi(i)$ und $c'_j = reenc(c_i))$, $reenc$ ist dabei die Wiederverschlüsselungsfunktion. Die Permutation π soll in dem Beweis geheim bleiben.

Der Mixserver geht dazu folgendermaßen vor:

1. Er erstellt k sogenannte *Shadow Mixes* $\mathcal{C}''_1, \ldots, \mathcal{C}''_k$ von $\mathcal{C}$. Dazu erstellt er k Paare von Permutationen $(\pi_{L,j}, \pi_{R,j})$ mit $\pi_{L,j} \circ \pi_{R,j} = \pi$, $j = 1, \ldots, k$. Für jeden Shadow Mix $\mathcal{C}''_j$ berechnet er $c''_{\pi_{L,j}(i),j} := reenc(c_i), i = 1, \ldots, n$, $\mathcal{C}''_j := \{c''_{1,j}, \ldots, c''_{n,j}\}$.

2. Der Mixserver veröffentlicht nun alle Shadow Mixes $\mathcal{C}''_1, \ldots, \mathcal{C}''_k$, hält aber die zugehörigen Permutationen $(\pi_{L,j}, \pi_{R,j}), j = 1, \ldots, k$ geheim.

3. Danach bekommt der Mixserver k Zufallsbits $b_1, \ldots, b_k$ (gewonnen z.B. mit der Fiat Shamir Heuristik aus Abschnitt 2.3.3) als Challenge: Ein Bit pro Shadow Mix.

4. Für jedes Bit b_j, $j = 1, \ldots, k$ tut er nun folgendes:

   - Falls $b_j = 0$: Der Mixserver legt $\pi_{L,j}$ offen und beweist, dass $c''_{\pi_{L,j}(i),j} = reenc(c_i)$ für alle $i = 1, \ldots, n$.
   - Falls $b_j = 1$: Der Mixserver legt $\pi_{R,j}$ offen und beweist, dass $c_{\pi_{R,j}(i)})' = reenc(c''_{i,j})$ für alle $i = 1, \ldots, n$. Diese Wiederverschlüsselung wurde zwar nicht konkret berechnet, kann aber vom Mixserver bewiesen werden, da er den jeweiligen Zufall kennt, der zur Wiederverschlüsselung $c''_{\pi_{L,j}(i),j} = reenc(c_i)$ und $c'_{\pi(i)} = reenc(c_i))$ verwendet wurde.

Der Mixserver führt damit einen Zero-Knowledge-Beweis für korrektes Mischen durch. Der geheimzuhaltende Zeuge ist dabei die Permutation π zusammen mit den zur Wiederverschlüsselung verwendeten Zufallswerten. In keinem der Beweisschritte wird offengelegt, welches Chiffrat aus $\mathcal{C}$ welchem aus $\mathcal{C}'$ entspricht. Genauer:

Completeness: Ein ehrlicher Mixserver kann die Schritte 1. bis 4. immer korrekt durchführen.

Soundness: Um etwas zu Manipulieren (Chiffrate aus $\mathcal{C}$ weglassen/duplizieren, falsche Chiffrate zu $\mathcal{C}'$ hinzunehmen), muss der Mixserver in jedem Shadow Mix $\mathcal{C}''_j$ auf einer Seite schummeln, so dass es jeweils ein i gibt für das entweder $c''_{\pi_{L,j}(i),j} \neq reenc(c_i)$ oder $c_{\pi_{R,j}(i)})' \neq reenc(c''_{i,j})$. Damit kann er eine der beiden Challenges in Schritt 4. nicht korrekt beantworten. Da der Mixserver vorher die Challenge nicht kennt, weiß er auch nicht, wo er schummeln muss, um nicht erwischt zu werden. Eine Manipulation wird also in jedem der k Durchläufe mit Wahrscheinlichkeit 50% entdeckt. Mit jedem weiteren Shadow Mix halbiert sich die Wahrscheinlichkeit, dass die Manipulation nicht entdeckt wird. Bei k Shadow Mixes wird damit eine Manipulation nur mit Wahrscheinlichkeit $\frac{1}{2^k}$ nicht erkannt.

Simulierbarkeit: Der Beweis ist simulierbar: Der Simulator wählt sich eine "Challenge" $b = b_1, \ldots, b_k$, erstellt entsprechend "Shadow Mixes" von $\mathcal{C}$ (für $b_j = 0$) oder $\mathcal{C}'$ (für $b_j = 1$), damit kann er dann immer richtig die simulierten $\pi_{L,j}$ bzw. $\pi_{R,j}$ aufdecken, ohne π zu kennen: er kennt die Zufallchallenge b vorher und weiss entsprechend, welche Permutationshälfte ($\pi_{L,j}$ oder $\pi_{R,j}$) er jeweils aufdecken muss. Genau diese Hälfte kann er sich jeweils konstruieren, ohne die zweite Hälfte berechnen zu können, d.h. er muss zu seinen $\pi_{R,j}$ kein $\pi_{L,j}$ kennen mit $\pi_{L,j} \circ \pi_{R,j} = \pi$ und umgekehrt.

2.5.2 Randomized Partial Checking

Randomized partial checking (RPC) ist um einiges effizienter als der oben beschriebene Re-Encryption-Mix mit Shadow Mixes. Es gilt wieder zu beweisen, dass eine Liste von n Chiffraten $\mathcal{C}' = \{c'_1, \ldots, c'_n\}$ ein *shuffle* einer Liste $\mathcal{C} = \{c_1, \ldots, c_n\}$ ist, d.h. es gibt für jedes $c_i \in \mathcal{C}$ genau ein $c'_j \in \mathcal{C}'$ mit $j = \pi(i)$ und $c'_j = reenc(c_i)$), $reenc$ ist wieder die Wiederverschlüsselungsfunktion und Verschlüsselung und Wiederverschlüsselung kann wieder bei Bedarf gedanklich durch Commitment und Maskierung ersetzt werden. Der Mixserver geht nun folgendermaßen vor:

1. Statt k Shadow Mixes erstellt er nur einen: Er wählt ein Permutationenpaar (π_L, π_R) mit $\pi_L \circ \pi_R = \pi$. Dann berechnet er $c''_{\pi_L(i)} := reenc(c_i), i = 1, \ldots, n$, $\mathcal{C}'' := \{c''_1, \ldots, c''_n\}$.

2. Der Mixserver veröffentlicht den Shadow Mix $\mathcal{C}''$, hält aber die zugehörigen Permutationen (π_L, π_R) geheim.

3. Danach bekommt der Mixserver n Zufallsbits $b_1, \ldots, b_n$ (gewonnen wieder z.B. mit der Fiat Shamir Heuristik aus Abschnitt 2.3.3) als Challenge: Diesmal ein Bit pro Chiffrat.

4. Für jedes Bit b_i, $i = 1, \ldots, n$ tut er nun folgendes:

   - Falls $b_i = 0$: Der Mixserver legt $\pi_L^{-1}(i)$ offen und beweist, dass $c''_i = reenc(c_{\pi_L^{-1}(i)})$.
   - Falls $b_i = 1$: Der Mixserver legt $\pi_R(i)$ offen und beweist, dass $c_{\pi_R(i)})' = reenc(c''_i)$.

RPC ist zwar viel effizienter als der Beweis mit mehreren Shadow-Mixes, dafür aber in der oben beschriebenen Variante unsicherer: Wie man sieht, wird nun von jedem Chiffrat c'' im Shadow Mix entweder die linke oder die rechte Zuordnung aufgedeckt. Dies verrät aber nun teilweise Information über π: Für jedes Paar (i, j) mit $b_i = 0$ und $b_j = 1$ sieht man, dass $c'_{\pi_R(j)}$ nicht eine Wiederverschlüsselung von $c_{\pi_L^{-1}(i)}$ sein kann, d.h. $\pi_R(j) \neq \pi(\pi_L^{-1}(i))$, denn da schon die Zuordnung zwischen c''_i und $c_{\pi_L^{-1}(i)}$ offengelegt wurde, wäre die entsprechende Zuordnung zwischen c''_i und $c'_{\pi_R(i)} = c'_{\pi(\pi_L^{-1}(i))}$ niemals aufgedeckt worden. Damit kann man für jedes Chiffrat in $\mathcal{C}$ etwa die Hälfte aller Chiffrate in $\mathcal{C}'$ als deren Wiederverschlüsselung ausschließen. Sind die Chiffrate in $\mathcal{C}$ Wählerstimmen, so "versteckt" sich die eigene Stimme immernoch in etwa der Hälfte aller Wählerstimmen, was bei einer hohen Anzahl Wählern immernoch gut genug sein kann.

Abhilfe gegen das Problem : mehrere Mixes (min. 4) nacheinander (TODO).

Kapitel 3

Erste Beispiele für kryptographische Wahlverfahren

Um einen Eindruck zu vermitteln, wie verifizierbare Wahlverfahren aussehen und mit welchen Bausteinen sie arbeiten, werden hier einige einfachere Wahlverfahren vorgestellt. Bei papierbasierten Wahlverfahren gibt der Wähler seine Stimme auf einem Papierstimmzettel ab. Es kann jedoch durchaus sein, dass die Stimme später elektronisch gezählt wird. Eine Sonderrolle spielen scannerbasierte Wahlverfahren, hier gibt der Wähler zwar seine Stimme auf Papier ab, jedoch wird der Stimmzettel danach eingescannt und die Stimme elektronisch gezählt. Da der Wähler seine Stimme auf Papier abgibt, und prinzipiell Papierstimmzettel normalerweise auch eingescannt werden können, siedeln wir diese Verfahren in dieser Vorlesung bei den papierbasierten Wahlverfahren an. Ein Beispiel für ein scannerbasiertes Wahlverfahren ist Scantegrity II, das in einem späteren Kapitel kurz vorgestellt wird. Bei elektronisch unterstützten Wahlverfahren gibt der Wähler seine Stimme an einem Wahlcomputer oder einer DRE (direct-recording electronic) ab.

3.1 Papierbasierte Präsenzwahlverfahren

Ein Beispiel für papierbasierte Wahlverfahren ist natürlich unsere Papierwahl. In diesem Abschnitt werden nun zwei weitere papierbasierte Wahlverfahren vorgestellt: Three Ballot, ein verifizierbares Wahlverfahren, das völlig ohne Kryptographie auskommt, und das Wahlverfahren Punchscan, bei dem der Wähler seine Stimme zwar auf Papier abgibt, diese jedoch computergestützt ausgezählt wird. Auch die Korrektheit der Auszählung wird bei Punchscan elektronisch bewiesen.

3.1.1 Punchscan

Punchscan [PH10] ist ein papierbasiertes Präsenzwahlverfahren, bei dem die Auszählung rechnergestützt erfolgt. Die Autorisierung der Wähler erfolgt wie bei herkömmlichen Wahlen. Das Ziel des Verfahrens ist es, dem Wähler zeigen zu können, dass / ob seine Stimme korrekt gezählt wurde, ohne Aufzudecken, für welchen Kandidaten er gestimmt hat. Dafür wird der Stimmzettel in zwei Teile aufgeteilt, die übereinander liegend zeigen, für welchen Kandidaten gestimmt wurde. Jeder Teil für sich, verrät jedoch nichts über die abgegebene Stimme.

Wie in Abbildung 3.1 zu erkennen ist, enthalten beide Teile des Stimmzettels eine ID, mit deren Hilfe sie einander eindeutig zuzuordnen sind. Auf dem vorderen Teil des Stimmzettels befinden sich außerdem die Liste der Kandidaten und eine Nummer zu jedem Kandidaten. Die Kandidaten sind auf jedem Stimmzettel in der gleichen Reihenfolge. Ihnen ist jedoch nicht immer die gleiche Nummer zugeordnet. Die Reihenfolge der Nummern ist zufällig und wird durch die Permutation π_{Top} beschrieben. Unten auf dem vorderen Teil des Stimmzettels befinden sich Aussparungen, so dass die auf dem hinteren Teil des Stimmzettels stehenden Nummern zu sehen sind (siehe Abb. 3.2). Die Nummern auf dem hinteren Teil des Stimmzettels sind ebenfalls in einer zufälligen Reihenfolge, beschrieben durch die Permutation π_{Bottom}.

3.1.1.1 Voraussetzungen

Wahlgeheimnis und Quittungsfreiheit:
Das Wahlverfahren setzt eine vertrauenswürdige Wahlauthority voraus. Unter dieser Voraussetzung bleibt das Wahlgeheimnis gewahrt und die Quittungsfreiheit ist erfüllt.

Nicht-Erpressbarkeit:
Wird nicht erfüllt.

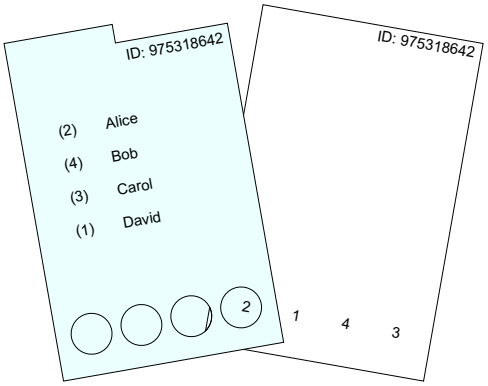

Abbildung 3.1: Punchscan-Stimmzettel – bestehend aus zwei Teilen

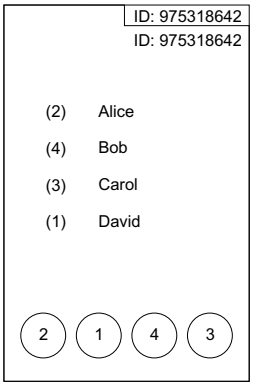

Abbildung 3.2: Punchscan-Stimmzettel – übereinander gelegt lassen sich die IDs vergleichen und die Nummern (1 – 4) auf dem hinteren Teil des Stimmzettels durch die Aussparungen des vorderen Teils lesen.

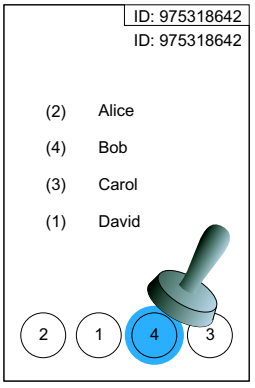

Abbildung 3.3: Wahl mit Punchscan-Stimmzettel – Der Wähler markiert mit einem Stempel seine Auswahl. Die Markierung ist auf beiden Stimmzettelhälften sichtbar, eine Hälfte allein verrät jedoch nicht die jeweils andere Permutation, damit bleibt das Wahlgeheimnis geschützt, solange man nur eine Stimmzettelhälfte sieht. Die Auswahl in diesem Beispiel entspricht einer Stimme für Bob.

Verifizierbarkeit:
Mehrere Auditoren (z.B. die Kandidaten), die als Gruppe vertrauenswürdig sind, für diverse Zufallschallenges.

3.1.1.2 Teilnehmer

Wähler Der Wähler gibt seine Stimme ab und überzeugt sich am Ende der Wahl davon, dass sein Stimmzettel gezählt wurde.

Kandidaten/Auditoren Die Auditoren erstellen gemeinsam Zufallschallenges für die Korrektheitsbeweise und überprüfen die korrekte Erstellung der Wahlzettel. Es wird für die Verifizierbarkeit davon ausgegangen, dass die Kandidaten wegen ihrer gegensätzlichen Interessen als Gruppe vertrauenswürdig sind und als Auditoren agieren können. Sie sehen keine Information, die das Wahlgeheimnis brechen könnte.

Wahlauthorität Die Wahlauthorität wird als Vertrauenswürdig vorausgesetzt und erstellt die Stimmzettel sowie alle zur späteren Auszählung und Verifizierung notwendigen Berechnungen und Korrektheitsbeweise. Sie führt auch die Auszählung am Ende der Wahl durch und veröffentlicht das Wahlergebnis zusammen mit den Korrektheitsbeweisen und allen für die Verifizierbarkeit notwendigen Daten.

Wahlorganisation Die Wahlorganisation führt die Wahl durch, prüft die Wahlberechtigung der Wähler und verteilt die Stimmzettel an die Wähler.

3.1.1.3 Wahlvorgang

Der Wähler gibt seine Stimme in einer Wahlkabine in einem ihm zugeordneten Wahlbüro ab. Dazu durchläuft er folgende Schritte:

1. Der Wähler geht ins Wahllokal und authentifiziert sich dort bei der Wahlorganisation als wahlberechtigter Wähler.
2. Der Wähler erhält von der Wahlorganisation einen Stimmzettel bestehend aus einer oberen und einer unteren Hälfte, wie in den Abbildungen 3.1, 3.2 und 3.3 abgebildet.
3. Der Wähler entscheidet sich, ob er später die obere oder die untere Hälfte des Stimmzettels als Quittung mitnehmen möchte, und teilt seine Entscheidung der Wahlorganisation mit.
4. Der Wähler begibt sich in die Wahlkabine und markiert auf dem Stimmzettel seine Auswahl. Dazu schaut er, welche Zahl neben seinem Kandidaten steht, und setzt mit einem Stempel

eine Markierung bei der Aussparung mit der entsprechenden Zahl so, dass die Markierung auf der oberen und unteren Hälfte des Stimmzettels zu sehen ist, wie in Abbildung 3.3 dargestellt.

5. Der Wähler trennt die obere Hälfte von der unteren. Die Hälfte, für die er sich bei Erhalt des Stimmzettels entschieden hat, wird eingescannt und für die Auszählung gespeichert, genau diese Hälfte darf der Wähler auch mit nach Hause nehmen zur Verifizierung seiner Stimmabgabe. Die andere Hälfte wird zerstört, da sie das Wahlgeheimnis brechen würde. Sie spielt keine weitere Rolle[1] mehr. Egal, welche Hälfte sich der Wähler aussucht, eine Hälfte allein verrät nichts über die Wählerstimme. Nur die Wahlauthority, die die Gesamtpermutation jedes Stimmzettels kennt, kann daraus die Stimme rekonstruieren.

6. Der Wähler nimmt die eingescannte Stimmzettelhälfte mit nach Hause. Nach der Wahl werden im Rahmen der Korrektheitsbeweise alle jeweils eingescannten Stimmzettelhäften (und nur diese!) auf einer Internetseite veröffentlicht, der Wähler kann nachprüfen, ob seine Stimmzettelhälfte dort zu finden ist und damit bei der Auszählung berücksichtigt wurde.

3.1.1.4 Vor der Wahl

Aus Wählersicht ist das Verfahren sehr einfach: Der Wähler markiert seine Auswahl entsprechend der Permutation auf dem Stimmzettel, nimmt eine Hälfte als Quittung mit und prüft nach, ob diese später auf einer Internetseite veroöffentlicht wurde. Schauen wir uns nun an, was im Hintergrund passiert.

Sei n die Anzahl Wähler. In der Preelectionphase werden $2n$ Stimmzettel erstellt (die Hälfte davon wird zu Auditzwecken verwendet, die andere Hälfte zum Wählen). Die Wahlauthorität wählt dazu pro Stimmzettel i die Seriennummer ID_i und zwei Permutationen $\pi_{i,Top}$ und $\pi_{i,Bottom}$, wodurch die beiden Stimmzettelhälften festliegen. Sie committet sich mit einer Commitmentfunktion com auf die Stimmzettel, indem sie für jeden Stimmzettel i $com(\pi_{i,Top})$ und $com(\pi_{i,Bottom})$ berechnet. Im Originalpapier wird ein Commitmentverfahren verwendet, das auf der AES-Verschlüsselung und der Hashfunktion SHA256 basiert, es können aber auch Pedersen-Commitments verwendet werden, sofern deren Parameter vertrauenswürdig (also von den Auditoren) gewählt werden, damit das Commitment für die Wahlauthority auch bindend ist. Die Wahlauthorität bereitet drei Tabellen vor, die im Laufe der Wahl vollständig ausgefüllt und teilweise veröffentlicht werden: Die Tabelle P (wie *Print*) für die Stimmzettel, die Tabelle D (wie *Decrypt*) für die teilweise geheime zur Auszählung benötigte Information, und die Tabelle R (wie *Result*) für das Wahlergebnis.

Die Tabelle P Die Tabelle P ist nach der Seriennummer der Stimmzettel sortiert und enthält jeweils eine Spalte P_1 für die obere Stimmzettelhälfte $\pi_{i,Top}$, eine Spalte P_2 für die untere Stimmzettelhälfte $\pi_{i,Bottom}$, sowie zwei Spalten für die commitments auf P_1 und P_2, also die Commitments $com(\pi_{i,Top})$ und $com(\pi_{i,Bottom})$. Die Spalten P_1 und P_2 selbst bleiben geheim, die Commitments werden veröffentlicht, in der richtigen Reihenfolge.

Die Tabelle D Die Tabelle D enthält wie P eine Zeile für jeden Stimmzettel, jedoch in permutierter Reihenfolge, so dass nicht ersichtlich ist, welche Zeile in D zu welcher Zeile in P gehört. In der Tabelle D steht für jeden Stimmzettel die Information, die zum Rekonstruieren der Wählerstimme aus einer Stimmzettelhälfte erforderlich ist. Dazu wählt die Wahlautorität für jeden Stimmzettel i zwei weitere Permutationen $\pi_{i,L}$ und $\pi_{i,R}$ mit $\pi_{i,L} \circ \pi_{i,R} = \pi_{i,Top} \circ \pi_{i,Bottom}$. Die Tabelle D enthält eine Spalte $P-Pointer$ mit je einem Pointer auf die entsprechende Zeile in P, damit legt die Spalte $P-Pointer$ eine Zeilenpermutation $\pi_{P-Pointer}$ fest. Genauso enthält sie eine Spalte $R-Pointer$ mit je einem Pointer auf die entsprechende Zeile in R, die eine Zeilenpermutation $\pi_{R-Pointer}$ festlegt. Daneben gibt es in D eine nach der Wahl auszufüllende, öffentliche Spalte für die Auswahl des Wählers v_i. Dabei ist v_i ein Vektor, der anzeigt, die wievielte Stelle auf der eingeworfenen Stimmzettelhälfte markiert wurde. Es spielt dabei keine Rolle, ob es sich um die obere oder untere Hälfte handelt, da zum Entschlüsseln der Stimme die Permutationen $\pi_{i,L}$ und $\pi_{i,R}$ direkt auf v_i angewendet werden. Es interessiert also nicht, welche Zahl angekreuzt wurde, sondern die wievielte Stelle, und diese Information ist auf beiden Stimmzettelhälften eindeutig erkennbar. Weiterhin enthält D eine Spalte $D3$ für die Permutationen $\pi_{i,L}$, eine Spalte $D4$ für die Permutationen $\pi_{i,R}$, und

[1] Die zweite Hälfte wird zur Auszählung nicht benötigt. Die Wahlauthority kennt die Gesamtpermutationen jedes Stimmzettels, daher ist für die Auszahlung nur wichtig, welche der Aussparungen markiert wurde, aber nicht, welche Zahl darinstand bzw. zu welchem Kandidaten die markierte Zahl gehört. Die Zahlen dienen nur dem Wähler als Orientierung, welche Aussparung er markieren muss.

eine nach der Wahl auszufüllende Spalte $D5$ für die teilentschlüsselte Stimme $\pi_{i,L}(v_i)$. Ausserdem enthält D je ein Commitment auf jede Zeile in D sowie Spalten für Commitments auf $\pi_{i,L}$ und den P-Pointer und Commitments auf $\pi_{i,R}$ und den R-Pointer. Die Spalten mit den Commitments werden veröffentlicht, alles andere (Pointer und Permutationen) bleibt geheim. Die Spalte mit der teilentschlüsselten Stimme wird veröffentlicht, sobald sie ausgefüllt wurde.

Die Tabelle R Die Tabelle R wird nach der Wahl ausgefüllt und enthält die entschlüsselten Klartextstimmen $V_i := (\pi_{i,L} \circ \pi_{i,R})(v_i)$ in entsprechend dem R-Pointer aus D permutierter Reihenfolge. Sie wird nach der Wahl im Klartext veröffentlicht, aus ihr kann das Wahlergebnis berechnet werden.

Vor der Wahl suchen sich die Auditoren gemeinsam zufällig die Hälfte der Stimmzettel aus. Die entsprechenden Zeilen in allen Tabellen werden komplett offengelegt (für alle sichtbar auf dem Bulletin Board) und aus den Tabellen genommen. (Die entsprechenden Stimmzettel dürfen nicht mehr zum Wählen verwendet werden, weil hier der Weg von der Wählerquittung zur Klartextstimme offengelegt wäre.) Die Auditoren überzeugen sich anhand der geöffneten Commitments davon, dass die Stimmzettel zusammen mit den Entschlüsselungsinformationen, bestehend aus den Permutationen $\pi_{i,L}$ und $\pi_{i,R}$ und den P- und R-Pointern der entsprechenden Zeilen, korrekt erstellt wurden. Sie überzeugen sich insbesondere davon, dass für jeden der geöffneten Stimmzettel die Relation $\pi_{i,L} \circ \pi_{i,R} = \pi_{i,Top} \circ \pi_{i,Bottom}$ gilt.

3.1.1.5 Nach der Wahl

Individuelle Verifizierbarkeit Nach der Wahl werden alle Quittungen veröffentlicht. Jeder Wähler kann überprüfen, dass seine Quittung veröffentlicht wurde.

Universelle Verifizierbarkeit Jeder kann nachprüfen, dass die veröffentlichten Stimmzettelhälften den v_i in der Tabelle D entsprechen. Für den Beweis der korrekten Auszählung wählen die Auditoren ein Zufallsbit b.

- Falls $b = 0$: Die Wahlautorität öffnet alle Commitments zu den Permutationen in Spalte $D3$, also die Permutationen $\pi_{i,L}$, zusammen mit dem P-Pointer, damit wird der Weg von der aus den jeweiligen veröffentlichten Stimmzettelhälften hervorgehenden Auswahl v_i zur teilentschlüsselten Stimme $\pi_{i,L}(v)$ offengelegt. Da der Weg von $\pi_{i,L}(v)$ zur Klartextstimme verschlossen bleibt, bleibt das Wahlgeheimnis gewahrt.

- Falls $b = 1$: Die andere Seite wird aufgedeckt: der Weg von der teilentschlüsselten Stimme $\pi_{i,L}(v)$ zur Klartextstimme in der Tabelle R. Dazu werden der R-Pointer und die Commitments zur Spallte $D4$ für die Permutationen $\pi_{i,R}$ aufgedeckt. Da diesmal der Weg von der Auswahl zur teilentschlüsselten Stimme verborgen bleibt, kann wieder keine Verbindung zwischen der Wählerquittung und der Klartextstimme hergestellt werden.

Mit diesem Vorgehen hat die Wahlautorität 50% Chance, das Wahlergebnis zu manipulieren. Um diese Wahrscheinlichkeit zu veringern, verwenden wir das gleiche Vorgehen, das wir schon von anderen Wahlverfahren kennen: Die Wahlautorität erstellt nicht nur eine Tabelle D, sondern k Instanzen $D^{(1)}, \ldots, D^{(k)}$ davon, wobei k eine ausreichend große Zahl ist, in der Praxis wird $k = 80$ als angemessen angesehen. Die Tabellen P und R bleiben gleich, d.h. man wählt für jede Instanz $D^{(j)}, j = 1, \ldots, k$ einen neuen P-Pointer, dieser bestimmt die Reihenfolge der Zeilen in $D^{(j)}$. Außerdem wählt man für jede Instanz $D^{(j)}$ neue Permutationen $\pi_{i,L}$ und $\pi_{i,R}$ mit $\pi_{i,L} \circ \pi_{i,R} = \pi_{i,Top} \circ \pi_{i,Bottom}$. Der R-Pointer wird entsprechend dem P-Pointer (also der Zeilenreihenfolge in $D^{(j)}$) und der Reihenfolge in der Tabelle R angepasst.

Die Zufalls-Challenge besteht nun aus k bit $b_1, \ldots, b_k$. Nun wird von jeder Instanz $D^{(j)}$ entsprechend dem Bit b_j wie oben beschrieben entweder der Weg von der Wählerauswahl zur teilentschlüsselten Stimme oder der Weg von der teilentschlüsselten Stimme zur Klartextstimme aufgedeckt. Der Wahlautorität bleibt so bei ehrlich (und nach der Erstellung der $D^{(1)}, \ldots, D^{(k)}$!!) gewähltem Zufall eine Chance, unentdeckt zu manipulieren, von $\frac{1}{2^k}$.

Wer möchte, kann sich nun überlegen, warum dies ein Zero-Knowledge-Beweis ist. Wie würde man den Beweis simulieren?

3.1.1.6 Stärken und Schwächen des Verfahrens

Benutzerfreundlichkeit Das Verfahren wirkt sehr benutzerfreundlich, da der Wähler wie gewohnt seine Stimme auf Papier abgeben kann, und keinen Wahlcomputer bedienen muss. Die Verifikation ist für jeden Wähler optional, es ist auch möglich, eine Kopie der Quittung an eine vertrauenswürdige Person oder Organisation weiterzugeben, die das Überprüfen übernimmt. (Ganz optional ist die Verifikation nicht, wenn die Wahlauthorität vorher weiß, wer seine Stimme nicht überprüft, z.B. weil die Quittung im Wahlbüro im Mülleimer liegt, kann sie diese Stimme manipulieren. Dies gilt übrigens auch für einige andere Wahlverfahren, z.B. Bingo Voting in der ursprünglichen Version ohne Hashkette.)

Die beiden Permutationen könnten jedoch für einige Wähler etwas verwirrend sein. Außerdem könnten diese mit gesetzlichen Bestimmungen im Widerspruch stehen, da oft die Kandidatenreihenfolge oder sogar das ganze Stimmzettellayout gesetzlich vorgegeben ist.

Sicherheit Da sowohl die Wählerquittung als auch die Korrektheitsbeweise, auch in Verbindung miteinander, keine Information über die Wählerstimme preisgeben, bleibt das Wahlgeheimnis erhalten und das Verfahren ist Quittungsfrei, d.h der Wähler kann dem Angreifer nicht beweisen, was er gewählt hat. Dennoch ist der Wähler in diesem Verfahren erpressbar. Der Angreifer kann den Wähler zwar nicht zwingen, etwas bestimmtes zu wählen, er kann ihn aber zwingen, statt seiner gewünschten Wahl eine zufällige Stimme abzugeben, in dem er vom Wähler verlangt, dass auf der Quittung z.B. das erste Feld markiert sein muss. Diesen Angriff kann man umgehen, indem man dem Wähler mehrere Stimmzettel zur Auswahl gibt, was allerdings in einer enormen Mehrproduktion von Stimmzetteln resultieren würde, da kein Wähler den Stimmzettel eines anderen sehen darf. Wie bei jeder Präsenzwahl ist eine erzwungene Enthaltung nicht auszuschließen, da man einfach beobachten kann, wer zur Wahl geht.

Ballot Stuffing muss durch organisatorische Maßnahmen garantiert werden, ansonsten ist die Verifizierbarkeit gegeben.

3.1.2 ThreeBallot

Das "ThreeBallot Voting System" [Riv06], entworfen von Ronald L. Rivest ist ein Papierwahlverfahren, dass, wie das z.Z. eingesetzte Wahlverfahren in Deutschland, ohne kryptographische Methoden auskommt. Es stellt einen Versuch dar, ob die Eigenschaften, dass ein Wahlverfahren sowohl geheim als auch für den einzelnen Wähler verifizierbar ist, miteinander vereinbar sind, ohne Zuhilfenahme kryptographischer Methoden.

Der Stimmzettel des Verfahrens besteht aus drei Teilen, wobei jeder Teil für sich einem kompletten Stimmzettel des herkömmlichen Verfahrens entspricht. Zusätzlich enthält jeder Teil des Stimmzettels noch eine zufällige ID. Das bedeutet, dass die drei Teile, nachdem sie getrennt und mit den Teilen aller anderen Stimmzettel gemischt wurden, einander nicht mehr zuzuordnen sind, aber jeder Teil für sich leicht wiedergefunden werden kann, wenn die ID bekannt ist.

3.1.2.1 Voraussetzungen

Wahlgeheimnis und Quittungsfreiheit Hierzu wird eine vertrauenswürdige Wahlauthorität vorausgesetzt, die IDs für die Stimmzettel ehrlich (zufällig) wählt. Wählt sie diese stattdessen auf bestimmte Weise, so kann sie, wie man weiter unten sieht, das Wahlgeheimnis und damit die Quittungsfreiheit brechen. Im Wahlprotokoll werden eine Checker-Machine und ein Drucker verwendet, diese müssen auch vertrauenswürdig sein.

Erpressbarkeit Bei vielen Kandidaten sind Pattern-Voting-Angriffe möglich, Three Ballot eignet sich also nur für Wahlen mit wenigen Kandidaten. Wie bei jeder Präsenzwahl ist hier die erzwungene Enthaltung möglich, sogar einfacher als sonst: Es wird im Verfahren eine Liste der Wähler veröffentlicht. Weitere Angriffe werden weiter unten beschrieben.

Verifizierbarkeit Voraussetzung für die Verifizierbarkeit ist die ehrliche Veröffentlichung der Liste der Wähler, die eine Stimme abgegeben haben, zum Verhindern von Ballot Stuffing.

3.1.2.2 Stimmabgabe

Bis auf die Stimmabgabe verläuft das Wahlverfahren wie bei der herkömmlichen Wahl. Die Stimmabgabe funktioniert wie folgt:

1. Der Wähler gibt jedem Kandidaten zunächst je ein Kreuz auf einem beliebigen der drei Teile.
2. Der Wähler gibt dem Kandidaten, für den er stimmen möchte, auf einem der beiden Teilen, die bei diesem Kandidaten noch nicht mit einem Kreuz versehen sind, ein weiteres Kreuz.
3. Der Wähler übergibt den Stimmzettel nun einer Maschine, die die Gültigkeit überprüft. (Ob max. ein Kandidat zwei Kreuze hat, jeder andere genau eins.)
4. Die Maschine markiert den Stimmzettel als gültig und trennt die drei Teile voneinander.
5. Der Wähler erhält die Kopie einer der Teile als Beleg. Welchen der drei Teile er sich kopieren lässt, wählt der Wähler selbst.
6. Der Wähler gibt die drei Originale ab und behält den Beleg für die Kontrollmöglichkeit.

Nach der Wahl werden alle Stimmzettelteile gemischt und veröffentlicht.

3.1.2.3 Stimmauszählung

Die Berechnung, wie viel Stimmen der Kandidaten hat, funktioniert für jeden nachvollziehbar wie folgt:

1. Es wird gezählt, wie viele Kreuze ein Kandidat erhalten hat.
2. Davon wird abgezogen, wie viele Stimmzettel (Stimmzettelteile / 3) abgegeben wurden.

Der Wähler erhält nun einen Hinweis, ob seine Stimme bei der Auszählung berücksichtigt wurde, indem er den Beleg, den er als Kopie eines Stimmzettelteils erhalten hat, unter den Veröffentlichungen sucht und die Verteilung der Kreuze überprüft. Ein Angreifer, der ohne zu wissen, welcher Teil als Belegt dient, versucht die Wahl zu fälschen, fällt mit einer Wahrscheinlichkeit von 1/3 pro änderung auf, vorausgesetzt jeder Wähler überprüft seinen Beleg. Mit steigender Zahl der Manipulationen, sinkt die Wahrscheinlichkeit für den Angreifer nicht entdeckt zu werden sehr schnell.

Der Beleg, den der Wähler erhalten hat, enthält keinen Hinweis darauf, welchen Kandidaten er gewählt hat, da bei jedem Kandidaten, unabhängig davon, ob für ihn gestimmt wurde oder nicht, ein Kreuz vorhanden sein kann, oder nicht.

3.1.2.4 Stärken und Schwächen des Verfahrens

Die Stärke des Verfahrens liegt eindeutig in der Einfachheit und der Möglichkeit für den Wähler zu überprüfen, ob seine Stimme gezählt wurde. Das Verfahren bietet dem Wähler allerdings keine Möglichkeit die Korrektheit der gesamten Wahl zu verifizieren. Die korrekte Auszählung eines Wahlbezirks muss genauso erfolgen, wie bisher. Angriffe auf das Verfahren sind ebenfalls noch möglich. So ist die Art und Weise, wie die Kontrolle der Gültigkeit des Stimmzettels umgesetzt wird, von großer Bedeutung: Erhält der Wähler seinen Stimmzettel nach der Kontrolle und nachdem er als gültig markiert wurde wieder, so lässt er sich ggf. im Nachhinein noch manipulieren. Außerdem darf es dem Wähler nicht möglich sein, mehr als einen der Stimmzettelteile zu kopieren. Der Kopierer darf sich aber auch nicht merken können, welcher Teil / welche ID als Beleg dient, da die anderen beiden Teile sonst vor der Veröffentlichung manipuliert werden können. Sowohl der Checker als auch der Kopierer müssen dementsprechend vertrauenswürdig sein, da sie die Stimme lernen, bzw. Informationen erhalten, wie die Stimme manipuliert werden kann. Die Erzeugung der Stimmzettel muss ebenfalls von einer vertrauenswürdigen Instanz übernommen werden, da es nahezu unmöglich ist, die IDs als unabhängig voneinander zu beweisen. Pattern Voting ist ebenfalls nicht gänzlich auszuschließen. Die Gefahr, mit den Kreuzen wiedererkennbare Muster erzeugen zu können, steigt je mehr Kandidaten zu Wahl stehen.

Diese und weitere Gedanken zu dem Verfahren bespricht Ronald Rivest in dem Paper [Riv06], in dem er das Verfahren beschreibt. Er macht sich auch Gedanken, welche Eigenschaften der Checker erfüllen muss, und ob es sinnvoll ist, den Kopierer für den Beleg darin zu integrieren. Auch die Folgen gestohlener Belegen oder hinzugefügte Stimmen, werden von ihm diskutiert. In diesem Sinne: "Grandma, did you really vote? Weren't you sick that day?"[Riv06].

3.2 Computerunterstützte Wahlverfahren

3.2.1 Mix-basiertes Verfahren von Moran und Naor

Das Wahlverfahren von Tal Moran und Moni Naor [MN06] ist eins der ersten kryptographischen Präsenzwahlverfahren, die dem Wähler zum Überprüfen seiner Stimme eine Quittung mitgeben, mit der der Wähler zwar seine Stimmabgabe überprüfen kann, einem Dritten aber nicht beweisen kann, was er gewählt hat. Erreicht wird dies durch Zero-Knowledge-Beweise: Jede Quittung enthält unter anderem ein Commitment auf den gewählten Kandidaten, diese Commitments werden nach der Wahl zur Auszählung über einen verifizierbaren Mix geöffnet. Die Wahlmaschine beweist dem Wähler mit einem Zero-Knowledge-Beweis, dass der von ihm gewählte Kandidat gezählt wurde, dass also auf seiner Quittung ein Commitment auf den gewählten Kandidaten steht. Zusätzlich bekommt der Wähler für alle anderen Kandidaten jeweils einen simulierten Zero-Knowledge-Beweis, dass das Commitment diesen Kandidaten enthält, die Stimmabgabe also für diesen Kandidaten war. Nur der Wähler kann zwischen dem echten und den simulierten Beweisen unterscheiden: Er hat in der Wahlkabine der Wahlmaschine die Zufallschallenge für die Beweise der nicht gewählten Kandidaten vor der Beweiserstellung verraten. Dadurch konnte die Wahlmaschine diese Beweise simulieren. Die Zufallschallenge für den gewählten Kandidaten bekam die Wahlmaschine jedoch erst zu einem späteren Zeitpunkt, so dass sie diesen Beweis ehrlich erstellen musste. Schauen wir uns nun das Verfahren im Detail an.

3.2.1.1 Voraussetzungen

Wahlgeheimnis, Quittungsfreiheit und Nicht-Erpressbarkeit Es wird eine vertrauenswürdige Wahlmaschine (bzw. eine vertrauenswürdige DRE) vorausgesetzt, da diese die Wählerstimme im Klartext erhält. Außerdem wird ein *Committing Printer* vorausgesetzt. Dieser enthält einen Sichtschutz, der Teile des Ausdrucks verdeckt. Dadurch kann der Wähler nachprüfen, dass etwas gedruckt wurde, ohne den Ausdruck zu sehen. Quittungsfreiheit und Nicht-Erpressbarkeit wird leider trotzdem nicht vollständig erfüllt, es gibt einen Angriff (*babbling attack*, mehr dazu unten), der allerdings nicht skaliert und deshalb keine allzu große Gefahr darstellen sollte.

Verifizierbarkeit Für die verifizierbare Korrektheit wird ein *öffentlicher Random Beacon* vorausgesetzt, also eine öffentliche Zufallsquelle, die bei jedem Zugriff neuen unvorhersehbaren, öffentlich sichtbaren Zufall produziert. Dieser wird bei der Auszählung als Challenge für den Beweis des korrekten Mischens der Stimmen verwendet. Außerdem muss das DLOG-Problem in der für die Pedersen-Commitments verwendeten Gruppe schwer sein, sonst sind die Commitments nicht bindend und die DRE kann manipulieren.

3.2.1.2 Teilnehmer

DRE Eine DRE ist normalerweise eine Art Wahlmaschine, die einen herausnehmbaren Speicher enthält, auf dem die Stimmen gespeichert werden. In diesem Wahlverfahren kann man sich einfach eine Wahlmaschine vorstellen. Die DRE muss zur Wahrung des Wahlgeheimnisses vertrauenswürdig sein, für die Verifizierbarkeit allerdings nicht, Manipulationen werden bemerkt. Die vertrauenswürdige DRE impliziert natürlich eine vertrauenswürdige Wahlauthority, die die DRE aufsetzt, in die Wahlkabine stellt u.s.w., die DRE zusammen mit der Wahlauthority werden hier jedoch als eine Einheit betrachtet, die Wahlauthority wird nicht als zusätzlicher Teilnehmer gelistet.

Committing Printer In der Wahlkabine befindet sich ein Committing Printer (Drucker mit Sichtschutz, siehe oben), auf dem die DRE Ausgaben machen kann.

Bulletin Board Wie bei fast jedem kryptographischen Wahlverfahren gibt es auch hier ein öffentliches Bulletin Board, das für alle sichtbar ist. Darauf werden Korrektheitsbeweise und das Ergebnis der Auszählung veröffentlicht.

Wähler Die Wähler geben in einer Wahlkabine an einer vertrauenswürdigen DRE ihre Stimme ab und prüfen nach der Wahl die Korrektheitsbeweise.

3.2.1.3 Vor der Wahl

Ein oft angepriesener Vorteil dieses Wahlverfahrens ist, dass es ohne jegliche Vorberechnungen auskommt, wodurch eine Wahl relativ spontan stattfinden kann. Dadurch gestaltet sich dieser Abschnitt sehr kurz und ist schon fast zu Ende. Vor der Wahl sollten allerdings Parameter g und h für die in der Wahlphase verwendeten Pedersen-Commitments (siehe Abschnitt 2.2.1) festgelegt werden, mit g, h Erzeuger einer multiplikativen zyklischen Gruppe G, so dass $log_g(h)$ der DRE/-Wahlauthority nicht bekannt ist. Ausserdem werden ein Sicherheitsparameter k sowie ein Verfahren zum eindeutigen Bestimmen eines Bitstrings der Länge k aus ASCII-Text festgelegt.

3.2.1.4 Wahlphase

Der Wahlvorgang ist in Abbildung 3.4 dargestellt und wird hier nun genauer beschrieben. Zur Wahl begibt sich der Wähler in eine Wahlkabine. Dort befindet sich für die Stimmabgabe eine vertrauenswürdige DRE und einen Drucker mit Sichtschutz (*Committing Printer*), der einen Teil des Ausdrucks verdeckt. In der Wahlphase werden Pedersen-Commitments verwendet. Dazu wurden vor der Wahl die Parameter g und h und ein Sicherheitsparameter k bestimmt. Sei n die Anzahl Kandidaten.

Der Wähler und die DRE führen nun folgende Schritte aus (die Numerierung der Schritte entspricht der in Abbildung 3.4, dadurch kann der Wahlvorgang direkt am Beispiel nachvollzogen werden):

1. Der Wähler entscheidet sich für einen Kandidaten C_W und wählt diesen an der DRE aus. Die DRE berechnet ein Pedersen-Commitment $v = com(C_W, r) = g^{C_W} h^r$ auf den gewählten Kandidaten C_W (im Beispiel Betty) mit Commitment-Zufall r, in unserem Beispiel ist $v = com(Betty, r)$. Ausserdem erstellt die DRE k Maskierungen $b_1, \ldots, b_k$ des Commitments v mit Maskierungszufall $r_{B,1}, \ldots, r_{B,k}$, dabei ist $b_i = vh^{r_{B,i}} = com(C_W, r + r_{B,i}), i = 1, \ldots, k$

2. Der Wähler schreibt hinter jeden nicht gewählten Kandidaten $C_j, j = 1, \ldots, W-1, W+1, \ldots, n$ jeweils einen von ihm gewählten Zufallsstring, im Beispiel bestehend aus mehreren zufälligen Worten. Die DRE macht daraus jeweils einen Bitstring $B_j, j = 1, \ldots, W-1, W+1, \ldots, n$ der Länge k (wie vor der Wahl festgelegt, dies ist später nachprüfbar). Für jeden nicht gewählten Kandidaten C_j trifft die DRE nun Vorbereitungen für die Simulation eines Zero-Knowledge-Beweises (jeweils mit entspr. Bitstring B_j als "Challenge") dafür, dass v ein Commitment auf C_j ist (was v natürlich nicht ist, es ist ja ein Commitment auf C_W). Dazu berechnet die DRE nun vermeintliche Maskierungen von v: Sei $B_{j,i}$ das i-te Bit der "Challenge"B_j. Für jedes Bit $B_{j,i}, i = 1, .., k$ wählt die DRE eine Zufallszahl $r_{j,i}$ und berechnet ein Commitment $c_{j,i}$ folgendermaßen:

   - **Für $B_{j,i} = 0$:** Die DRE berechnet statt einer Maskierung von v ein neues Commitment auf den Kandidaten C_j mit Commitment-Zufall $r + r_{j,i}$:
   $$c_{j,i} = com(C_j, r + r_{j,i}) = g^{C_j} h^{r + r_{j,i}}.$$
   - **Für $B_{j,i} = 1$:** Die DRE maskiert tatsächlich das Commitment v auf den gewählten Kandidaten mit Maskierungszufall $r_{j,i}$:
   $$c_{j,i} = vh^{r_{j,i}} = com(C_W, r + r_{j,i}).$$

   Die DRE berechnet nun noch ein Commitment X auf alle Commitments, die sie bis dahin berechnet hat: Das Commitment v auf den gewählten Kandidaten, die Commitments $c_{j,i}$ für die Beweissimulation und die "richtigen" Maskierungen b_i von v für den "richtigen" Beweis. Dabei soll der Angreifer später nicht zwischen b_i und $c_{j,i}$ unterscheiden können, deshalb werden die b_i an der Stelle W des gewählten Kandidaten eingeordnet:
   $$All = v||c_{1,1}||\_||c_{1,k}|| \ldots ||c_{W-1,1}||\_||c_{W-1,k}||b_1||\_||b_k||c_{W+1,1}||\_||c_{W+1,k}|| \ldots ||c_{n,1}||\_||c_{n,k},$$
   $$X = com(All, r_x)$$
   dabei ist r_x der Commitmentzufall. Im Beispiel wäre hier an erster Stelle das Commitment v auf Betty, die Commitments $c_{1,1}, \ldots, c_{1,k}$ wären etwa zur Hälfte Commitments auf Alice und etwa zur Hälfte Commitments auf Betty, dann folgten die Commitments $b_1, \ldots, b_k$ auf Betty und danach die Commitments $c_{2,1}, \ldots, c_{2,k}$, die wiederum etwa zur Hälfte Commitments auf Charlie und etwa zur Hälfte Commitments auf Betty wären.

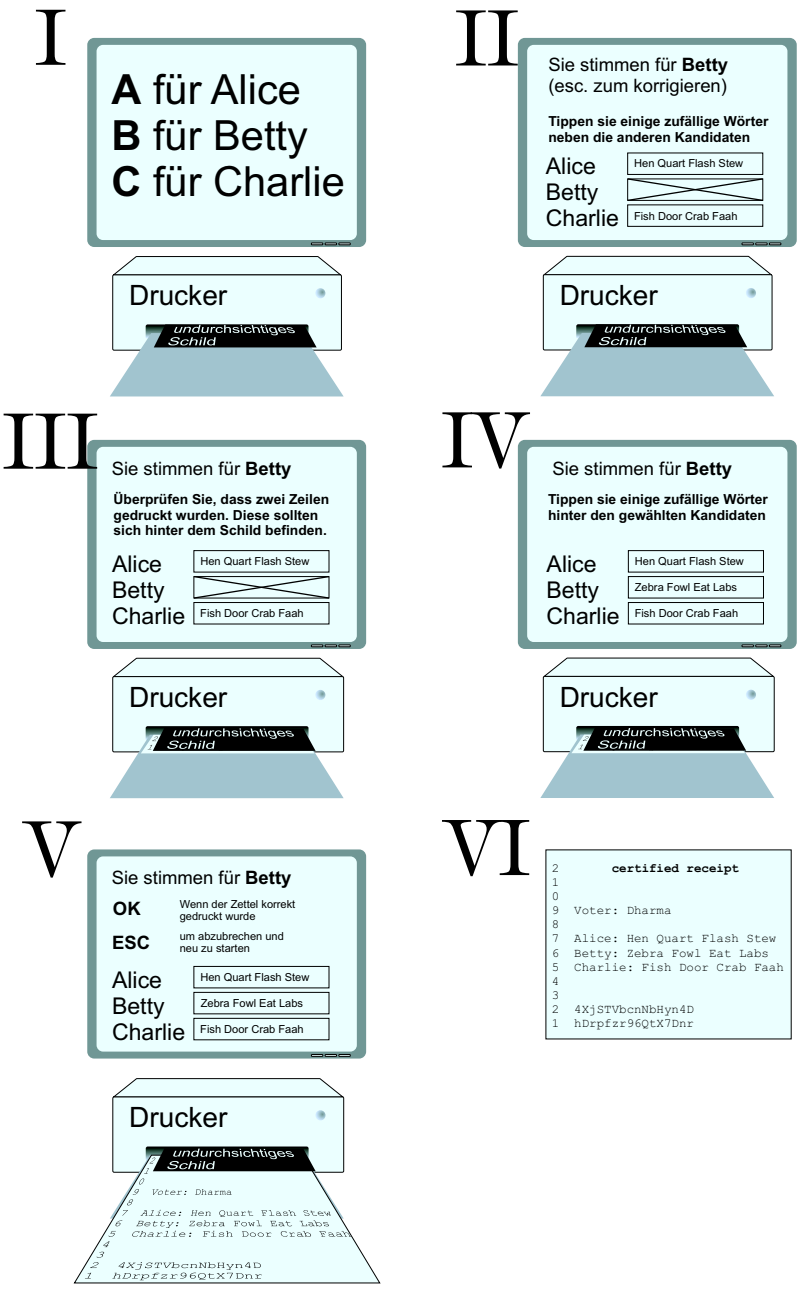

Abbildung 3.4: Stimmabgabe beim Wahlverfahren von Moran und Naor wie in [MN06] dargestellt

3. Das Commitment X wird nun auf dem Committing Printer ausgedruckt, bleibt aber noch verdeckt. Damit enthält die Wählerquittung nun implizit auch das Commitment v auf den gewählten Kandidaten. Der Wähler prüft, dass der Committing Printer etwa zwei Zeilen gedruckt hat, ohne zu sehen, was gedruckt wurde. Hat der Drucker nichts gedruckt, bricht der Wähler ab und startet von vorne.

4. Der Wähler gibt die Zufallschallenge für den gewählten Kandidaten ein, wie gehabt in Form von zufälligen Worten. Die DRE übersetzt dies wieder wie vorher festgelegt in einen k-Bitstring B_W, dieser wird später als "richtige" Challenge für den Zero-Knowledge-Beweis verwendet.

5. Der Drucker druckt nun den Rest der Quittung, so, dass nichts verdeckt bleibt. Auf der Quittung sollten nun der Name des Wählers, die Kandidaten mit den richtigen Zufallsworten dahinter und das Commitment X stehen. Der Wähler überprüft dies. Ist er unzufrieden oder möchte sich noch umentscheiden, so bricht er den Wahlvorgang ab und startet von vorne. Die DRE führt nun den echten und die simulierten ZK-Beweise zu Ende durch, indem sie die Antworten auf die Zufallschallenges berechnet. Sei B_j wieder die Zufallschallenge des Kandidaten C_j, und $B_{j,i}$ das i-te Bit von B_j. Erinnerung: Für $j = W$ ist $C_j = C_W$ der gewählte Kandidat und $B_j = B_W$ die "richtige" Challenge. Für alle Kandidaten $C_j, j = 1, \ldots, n$ (und damit einschliesslich C_W) "beweist" die DRE nun, dass v ein Commitment auf C_j ist, indem sie je nach Challengebit "zeigt", dass die $c_{j,1}, \ldots, c_{j,k}$ Maskierungen von v und gleichzeitig Commitments auf C_j sind. Seien dazu die $c_{W,i}$ die in Schritt 1 berechneten "echten" Maskierungen von v: $c_{W,i} = b_i, i = 1, \ldots, k$. Für jeden Kandidaten C_j für jedes Bit der Zufallschallenge B_j tut die DRE dazu folgendes:

   - **Falls** $B_{j,i} = 0$**:** Die DRE muss beweisen, dass $c_{j,i}$ ein Commitment auf C_j ist. Sie öffnet dazu das Commitment $c_{j,i}$, indem sie den Zufall $r + r_{j,i}$ offenlegt. Für $j = W$ ist $c_{j,i}$ immer ein Commitment auf C_j. Für $j \neq W$ ist $c_{j,i}$ genau dann ein Commitment auf C_j, wenn $B_{j,i} = 0$, da die DRE in diesem Fall in Schritt 2 bereits die Challenge kannte und statt einer Maskierung von v ein Commitment auf C_j erstellt hat. Damit kann sie nun immer korrekt aufdecken, dass $c_{j,i}$ ein Commitment auf C_j ist, ätsch.
   - **Falls** $B_{j,i} = 1$**:** Die DRE muss beweisen, dass $c_{j,i}$ eine Maskierung des Commitments v auf den gewählten Kandidaten ist. Dazu öffnet sie den Maskierungszufall $r_{j,i}$. Für $j = W$ ist $c_{j,i}$ immer eine Maskierung von v. Für $j \neq W$ ist $c_{j,i}$ genau dann eine Maskierung von v, wenn $B_{j,i} = 1$, da die DRE in diesem Fall in Schritt 2 bereits die Challenge kannte, und brav v maskiert hat, in dem Wissen, dass sie $c_{j,i}$ niemals öffnen muss. Damit kann sie immer korrekt aufdecken, dass $c_{j,i}$ eine Maskierung von v ist.

6. Nachdem der Wähler seine Quittung überprüft und bestätigt hat, gilt seine Stimme als gezählt, und "Certified Receipt" auf die Quittung gedruckt. Der Wähler kann nun die Quittung herausnehmen (es muss sichergestellt werden, dass er das vorher nicht kann), und mit nach Hause nehmen. Die DRE veröffentlicht die Quittung sowie die simulierten und echten ZK-Beweise auf dem Bulletin Board.

Wie man sieht, sieht man der Quittung nicht an, welcher Kandidat gewählt wurde. Da Pedersen-Commitments unconditionally hiding sind, würde man das der Quittung auch dann noch nicht ansehen, wenn das DLOG-Problem gelöst wäre. Der Wähler weiß zwar, welcher der zu seiner Quittung gehörenden ZK-Beweise simuliert sind, und welches der echte Beweis ist, weil er weiß, welche Zufallsworte er erst später eingegeben hat, nachdem das Commitment gedruckt wurde, er kann dies dem Angreifer aber nicht beweisen, da er über die Reihenfolge der Eingabe lügen kann (außer beim *babbling attack*, siehe unten). Dabei ist wichtig, dass der Wähler in Schritt 3 und 4 nur sieht, *dass* etwas gedruckt wurde, aber nicht *was*, denn sonst könnte der Angreifer ihn zwingen, für den gewählten Kandidaten vom Commitment X abhängigen Zufall einzugeben.

3.2.1.5 Nach der Wahl

Die Quittungen und die in der Wahlphase erstellten ZK-Beweise und Simulationen sind bereits auf dem Bulletin Board veröffentlicht. Nun werden noch die Commitments X auf jeder Quittung geöffnet, damit kann jeder nachprüfen, dass sich die DRE bereits vor der Eingabe der echten Challenge auf v und die in den Beweisen verwendeten $c_{j,i}$ festgelegt hat. Jeder Wähler kann nachprüfen, dass

seine Quittung online sichtbar ist und jeder kann prüfen, dass die Beweise zu den jeweiligen (auf den Quittungen sichtbaren) Challenges passen.

Um die Stimmen auszuzählen, öffnet die DRE nun noch über einen verifizierbaren Mix die Commitments v jedes Wählers. Da es dieser Commitments mehrere gibt, nennen wir sie ab hier $v_1, \ldots, v_l$, wobei l die Anzahl Wähler ist. Dabei lernen wir nun den im Originalpapier[MN06] verwendeten Zero-Knowledge-Beweis für korrektes Mischen kennen, der ähnlich funktioniert wie der uns bereits bekannte Beweis mit Shadow Mixes (siehe Abschnitt 2.5.1).

Die DRE führt dazu die folgenden Schritte k mal aus (sie erstellt also k Instanzen des ZK-Beweises):

1. Die DRE permutiert und maskiert die Commitments $v_1, \ldots, v_l$, mit einer Permutation π und Maskierungszufall $m_1, \ldots, m_k$ und erhält damit die Maskierungen $v'_{\pi(i)} = v_i h_{m_i}$. Die $v_1, \ldots, v_l$ sind bereits veröffentlicht, die $v'_{\pi(i)}, i = 1, \ldots, l$ werden nun auch veröffentlicht. Die Permutation π und die Maskierungszufälle $m_1, \ldots, m_k$ müssen geheim bleiben.

2. Die DRE bekommt vom *Random Beacon* ein Zufallsbit b.
   - **Falls** $b = 0$**:** Die DRE öffnet die Permutation π und alle Maskierungszufälle $m_1, \ldots, m_k$, damit zeigt sie, dass sie korrekt gemischt hat, und keine Commitments/Stimmen hinzugefügt, weggenommen oder verändert wurden.
   - **Falls** $b = 1$**:** Die DRE öffnet die Commitments $v'_{\pi(i)}, i = 1, \ldots, l$ zu den Klartextstimmen.

Die Schritte 1-2 werden k mal ausgeführt. So wird in etwa der Hälfte der Fälle ($b = 0$) das korrekte Mischen bewiesen, man kann also davon ausgehen, dass in der anderen Hälfte auch korrekt gemischt wurde. In den anderen Fällen ($b = 1$) wurden die Stimmen aufgedeckt. Sind das in jedem Durchlauf die selben Klartextstimmen, so kann man davon ausgehen, dass hier nichts manipuliert wurde. Dieser Beweis wird auf dem Bulletin Board veröffentlicht und kann von jedem nachgeprüft werden. In der Realität wird man zuerst Schritt 1 k mal ausführen, also alle Permutationen und Maskierungen erstellen. Danach wird man statt dem Random Beacon den Hashwert einer kryptographischen Hashfunktion H über einen bis dahin unbekannten Klartext, z.B. über alle Maskierungen $v'_{\pi(1)}, \ldots, v'_{\pi(l)}$ aller k Runden als Challengebits verwenden. Ist H eine geeignete kollisionsresistente kryptographischen Hashfunktion, so ist es für die DRE schwer, gleichzeitig die richtigen $v'_{\pi(i)}$ zu manipulieren und so zu wählen, dass der Hashwert eine Zufallschallenge liefert, bei der er genau dann die Klartete aufdecken muss, wenn sie manipuliert sind, und genau dann die Maskierungen, wenn sie nicht manipuliert sind. Die Klartextstimmen können nun gezählt und damit das Wahlergebnis berechnet werden.

3.2.1.6 Vor- und Nachteile des Verfahrens

Sicherheit Da alle Commitments unconditionally hiding und alle Beweise ZK sind, bleibt das Wahlgeheimnis geschützt, bis auf einen Angriff: Angenommen, der Angreifer hat während der Wahlphase eine Tonverbindung zum Wähler, z.B. über einen Knopf im Ohr. Dann kann er dem Wähler während der Wahl Challenges diktieren, die jeweils so lang sind, dass der Wähler sich diese nicht merken kann und damit gezwungen ist, sie in der Reihenfolge einzugeben, in der der Angreifer sie vordiktiert, insbesondere muss er die letzte vordiktierte Challenge beim gewählten Kandidaten eingeben. Damit weiss der Angreifer, welches die Challenge für den gewählten Kandidaten ist, der Wähler wäre erpressbar und sogar die Quittungsfreiheit und das Wahlgeheimnis wären gebrochen. Da dieser Angriff jedoch schwer großflächig durchführbar ist, könnte man argumentieren, dass er keine allzuhohe Gefahr darstellt.

Da die Namen der Wähler auf den Quittungen stehen, kann zwar Ballot Stuffing entgegengewirkt werden, dafür wird aber eine erzwungene Enthaltung leichter. (Da es sich um ein Präsenzwahlverfahren handelt, ist eine erzwungene Enthaltung sowieso schwer zu verhindern.)

Benutzerfreundlichkeit Dass der Wähler selbst Zufall eingeben muss, ist ein großer Nachteil des Verfahrens. Der Wähler muss überprüfen, dass der Drucker etwas gedruckt hat und dies bestätigen, das ist aber keine allzugroße Herausforderung. Er muss außerdem selbst prüfen, dass die richtige (später eingegebene) Challenge an der richtigen Stelle auf der Quittung steht. Die Überprüfung, dass die eigene Quittung online ist, kann auch durch eine dritte Person/Organisation erfolgen, die vom Wähler eine Kopie der Quittung bekommt. Auch die ZK-Beweise der einzelnen Quittungen können von dritten oder unabhängigen Organisationen übernommen werden. Um sich der Korrektheit sicher zu sein, sollten allerdings insgesamt die ZK-Beweise zu allen Quittungen geprüft

werden. Wenn die DRE weiß, welche Quittungen nicht überprüft werden, z.B. weil diese Quittungen im Mülleimer liegen oder direkt nach der Wahl (ohne Umweg an einem Kopierer vorbei) dem Angreifer übergeben werden, kann die DRE an dieser Stelle manipulieren. Dieses Problem haben auch andere Wahlverfahren wie z.B. Bingo Voting. Es kann verhindert werden durch eine sogenante *Hashkette* über alle Quittungen, dies wird beim Verfahren Bingo Voting genauer erklärt.

3.2.2 Wahlverfahren mit homomorpher Stimmauszählung von Cramer, Gennaro und Schoenmakers

Das Verfahren von Cramer, Gennaro und Schoenmakers [CGS97] ist eins der ersten Wahlverfahren, die als effizientere Alternative zum verifizierbaren Mischen eine homomorphe Stimmverschlüsselung benutzen: Die Stimmen werden mit einem additiv homomorphen Verfahren verschlüsselt. Die Auszählung kann dann direkt auf den Chiffraten berechnet werden, und nur das Ergebnis, die Summe der Stimmen, wird entschlüsselt. Dadurch, dass einzelne Stimmen nie entschlüsselt werden, spart man sich den Mischvorgang mitsamt der zugehörigen Zero-Knowledge-Beweise. Allerdings bringt gerade die Tatsache, dass die Stimmen nie entschlüsselt werden, wiederum einige Gefahren mit sich: Die Stimmen werden nach der Wahl einfach aufsummiert, d.h. es muss vor der Summenbildung sichergestellt sein, dass jeder Wähler tatsächlich eine gültige Stimme abgibt. Ansonsten könnte der Wähler durch Ausnutzen der Homomorphieeigenschaft einem Kandidaten etwa -5 oder 3 Stimmen geben, er verschlüsselt dazu einfach eine -5 oder 3 statt einer 0 oder 1, oder verdreifacht ein Chiffrat der 1. Dadurch wäre die Gleichheit der Wahl nicht mehr gewährleistet. Das muss nichteinmal absichtlich passieren, jedes fehlerhafte Chiffrat würde das Ergebnis verfälschen. Hierfür wird in [CGS97] ein Zero-Knowledge-Beweis vorgeschlagen, der beweist, dass in einem Chiffrat eine gültige Stimme steckt, ohne etwas über die Stimme zu verraten.

Das Verfahren legt seinen Schwerpunkt auf die homomorphen Auszählung und den Beweis der Gültigkeit der Stimme. Es wird nicht genauer spezifiziert, an welchem Gerät die Stimme abgegeben oder verschlüsselt wird. Es wird auch nicht festgelegt, wie die Wahlberechtigung des Wählers geprüft wird, oder wie geprüft werden kann, dass nur Stimmen von Wahlberechtigten Wählern ins Wahlergebnis einfließen.

Die genaue Einsatzumgebung des Verfahrens ist nicht spezifiziert, es kann als Internetwahlverfahren oder Präsenzwahlverfahren verwendet werden. In jedem Fall muss es noch erweitert werden um zusätzliche Maßnahmen, einige davon sind in den Voraussetzungen erwähnt.

3.2.2.1 Voraussetzungen

Das Verfahren ist für 1-aus-2-Wahlen konzipiert, also für zwei Kandidaten und eine Stimme pro Wähler, z.B. Ja/Nein-Wahlen. Bei diesem Verfahren wird keine vertrauenswürdige Instanz vorausgesetzt, allerdings wird nicht genauer spezifiziert, auf welchem Gerät die Stimme verschlüsselt wird. Dieses Gerät müsste dann doch als vertrauenswürdig angenommen werden, sofern der Stimmabgabevorgang nicht um Code Voting (siehe Abschnitt 6.2) oder ähnliches erweitert wird.

Wahlgeheimnis, Quittungsfreiheit und Nicht-Erpressbarkeit Da Stimmen mit ElGamal verschlüsselt werden, wird das DLog-Problem als schwer vorausgesetzt. Außerdem muss das Gerät, an dem die Stimme eingegeben wird (z.B. der Wähler-PC), vertrauenswürdig sein, sofern es die Stimme im Klartext lernt. Die Stimmentschlüsselung muss auf ausreichend viele Server verteilt sein, von denen nie mehr als eine gewisse Anzahl korrumpiert sind und zusammenarbeiten.

Verifizierbarkeit Das Gerät, das die Stimmverschlüsselung vornimmt, muss vertrauenswürdig sein, es sei denn, es werden zusätzliche Maßnahmen ergriffen, um die korrekte Verschlüsselung zu testen. Außerdem muss zusätzlich garantiert werden, dass nur Stimmen von wahlberechtigten Personen in das Wahlergebnis einfließen, und von jedem nur eine.

3.2.2.2 Teilnehmer

Wähler Verschlüsseln ihre Stimme mit ElGamal, beweisen, dass das Chiffrat eine gültige Stimme enthält und und geben das Chiffrat ab.

Authorities Menge von k Instanzen $A_1, \ldots, A_k$, die als Gruppe vertrauenswürdig sind. Sie entschlüsseln die Summe der Stimmen gemeinsam.

Bulletin Board Unter dem Bulletin Board kann man sich eine Webseite im Internet vorstellen. Es hat folgende Eigenschaften:

1. Jeder kann es lesen.
2. Jeder Wahlteilnehmer kann Nachrichten anhängen.
3. Alle Nachrichten werden von dem Teilnehmer, der sie anhängt, digital signiert.
4. Niemand kann etwas löschen.

3.2.2.3 Vor der Wahl

Die Authorities führen ein Schlüsselerzeugungsprotokoll für ein Threshold-ElGamal-Verschlüsselungsverfahren aus, siehe z.B. Kapitel 2.4.4.2. Dadurch erhalten sie einen öffentlichen Schlüssel pk, mit dem später die Wählerstimme verschlüsselt wird, und jede Authority A_i hat ein Share sk_i des zugehörigen geheimen Schlüssels sk. Das Transkript der Schlüsselerzeugung wird auf dem Bulletin Board veröffentlicht, bzw. die Authorities kommunizieren zur gemeinsamen Schlüsselerzeugung über das Bulletin Board. Dadurch kann jeder nachprüfen, dass wirklich alle Authorities an der Schlüsselerzeugung beteiligt waren. Durch die Schlüsselerzeugung und das Transkript erhält niemand Information über sk, und das jeweilige Share sk_i lernt niemand außer der entsprechenden Authority A_i. Zur Erinnerung: Mit Threshold-ElGamal können die Authorities gemeinsam entschlüsseln, ohne das Geheimnis sk zu rekonstruieren, d.h. selbst nach einem erfolgreichen Entschlüsselungsvorgang ist eine einzelne beteiligte Authority weiterhin nicht in der Lage, allein weitere Chiffrate zu entschlüsseln.

Außerdem bekommt jeder Wähler Schlüssel zum Signieren seiner Nachrichten. Wie genau signiert wird, und wie und von wem die Signaturschlüssel erstellt und verteilt werden, wird im Originalpapier [CGS97] nicht genauer spezifiziert.

3.2.2.4 Wahlphase

Der Wähler kann nur zwischen zwei Möglichkeiten wählen, es handelt sich also z.B. um eine Ja/Nein-Stimme. Dabei wird eine Auswahl, die wir im weiteren Verlauf die *Ja-Stimme* nennen, durch eine 1 symbolisiert, die andere, wir nennen sie die *Nein-Stimme*, durch eine -1. Verschlüsselt wird die Wählerstimme mit exponential ElGamal (siehe Abschnitt 2.4.4.4), mit dem vor der Wahl erzeugten öffentlichen Schlüssel pk des Threshold-ElGamal-Verfahrens. Sei dazu G_q die verwendete Gruppe und g und G Erzeuger von G_q. Für eine Ja-Stimme wird nun, da wir exponentielles ElGamal verwenden, $m_1 := G^1$ verschlüsselt, und für eine Nein-Stimme $m_0 := G^{-1}$. In der weiteren Beschreibung meinen wir mit "Es wird $1/-1$ verschlüsselt" und "Es wird G^1/G^{-1} verschlüsselt" das gleiche: im ersten Fall wird 1 oder -1 mit exponential ElGamal verschlüsselt, im zweiten Fall G^1 bzw. GH^{-1} mit "normalem" ElGamal, was auf's Gleiche rauskommt (man lasse sich bitte durch die zwei verwendeten Erzeuger nicht weiter irritieren).

Nun zum Wahlvorgang:

1. Der Wähler verschlüsselt seine Auswahl mit ElGamal, er entscheidet sich also für ein $v \in \{1, -1\}$ und verschlüsselt v. Dadurch erhält er das Chiffrat $c = (g^r, pk^r g^v)$, wobei r der Verschlüsselungszufall ist. Das Chiffrat c wird ab nun als *Stimmzettel* bezeichnet. *Bemerkung: Wir beschreiben hier eine einfachere Variante des Verfahrens, im Originalpapier verschlüsselt der Wähler hier nicht direkt v sondern einen Zufallswert $b \in \{1, -1\}$, und zur Stimmabgabe gibt er dann zusätzlich ein e ab mit $v = be$.*
2. Der Wähler muss nun in den nächsten Schritten die Gültigkeit seines Stimmzettels beweisen, indem er beweist, dass das Chiffrat c einen Wert aus $\{1, -1\}$ enthält, ohne Information über v zu verraten. Der Wähler berechnet dazu Werte x, y, a_1, b_1, a_2, b_2 folgendermaßen:
   - **Falls v=1:** Der Wähler wählt $\alpha, w, r_1, d_1 \in \mathbb{Z}_q$ zufällig. Danach berechnet er $x := g^\alpha$, $y := pk^\alpha G$, $a_1 := g^{r_1} x^{d_1}$, $b_1 := pk^{r_1}(yG)^{d_1}$, $a_2 := g^w$, $b_2 := pk^w$.
   - **Falls v=-1:** Der Wähler wählt nun $\alpha, w, r_2, d_2 \in \mathbb{Z}_q$ zufällig. Dann berechnet er $x := g^\alpha$, $y := pk^\alpha G^{-1}$, $a_1 := g^w$, $b_1 := pk^w$, $a_2 := g^{r_2} x^{d_2}$, $b_2 := pk^{r_2}(yG^{-1})^{d_2}$.
3. Der Wähler schickt x, y, a_1, b_1, a_2, b_2 an den Verifizierer (bzw. veröffentlicht die Werte auf dem Bulletin Board).

4. Nun erhält der Wähler ein zufälliges $c \in \mathbb{Z}_q$ als Zufallschallenge. Wo dieser Zufall herkommt, ist nicht genauer spezifiziert, es wird vorgeschlagen, ähnlich wie beim Verfahren von Moran und Naor einen Random Beacon zu verwenden oder eine kryptographische Hashfunktion auf des Wählers Eingabedaten x, y, a_1, b_1, a_2, b_2, um den Beweis nicht-Interaktiv zu machen.

5. Der Wähler beantwortet die Challenge:
   - **Falls v=1:** Der Wähler berechnet $d_2 := c - d_1$ und $r_2 := w - \alpha d_2$, d_1 und r_1 hatte er ja vorher zufällig gewählt.
   - **Falls v=-1:** Der Wähler berechnet $d_1 := c - d_2$ und $r_1 := w - \alpha d_1$, diesmal hatte er ja d_2 und r_2 zuvor zufällig gewählt.

   Der Wähler schickt d_1, d_2, r_1, r_2 an den Verifizierer bzw. veröffentlicht es auf dem Bulletin Board.

6. Der Verifizierer prüft, ob folgende Gleichungen erfüllt sind:
   $c = d_1 + d_2$
   $a_1 = g^{r_1} x^{d_1}$
   $b_1 = pk^{r_1}(yG)^{d_1}$
   $a_2 = g^{r_2} x^{d_2}$
   $b_2 = pk^{r_2}(yG^{-1})^{d_2}$.

3.2.2.5 Nach der Wahl

Nach der Wahl prüfen die Authorities die Gültigkeit aller abgegebenen Stimmen, indem sie die vom Wähler durchgeführten Beweise prüfen. Sei jeweils (A_i, B_i) die Stimme von Wähler i, $i = 1, \ldots, l$, l sei die Anzahl Wähler. Nun wird das Wahlergebnis berechnet: Zuerst werden die Wählerstimmen multipliziert:

$$(X, Y) := (\prod_{i=1}^{l} A_i, \prod_{i=1}^{l} B_i).$$

Das Ergebnis wird von den Authorities nach dem Threshold-ElGamal-Verfahren gemeinsam entschlüsselt zu G^T, wobei T das Wahlergebnis ist. Dabei wird auch das korrekte Entschlüsseln bewiesen. *Anmerkung: Da exponentielles ElGamal verwendet wurde, um die Verschlüsselung additiv homomorph zu machen, ergibt die Entschlüsselung nicht direkt die Auszählung T.* Jetzt muss nur noch der diskrete Logarithmus T von G^T berechnet werden, und dann kann das Wahlergebnis veröffentlicht werden. Da es sich um einen kleinen Klartextraum handelt, ist die Berechnung von T aus G^T effizient möglich, man berechne z.B. G^{-l} und multipliziere solange mit G, bis G^T rauskommt, Aufwand $O(l)$, also linear in der Anzahl Wähler.

3.2.2.6 Vor- und Nachteile des Verfahrens

Die Berechnung der Auszählung des Verfahrens ist wesentlich effizienter als eine Auszählung via Mix. Es kann außerdem auf eine 1-aus-m-Wahl erweitert werden, wie, das steht in [CGS97] beschrieben.

Ein paar Ecken und Kanten hat das Verfahren jedoch noch: Da der Wähler seine Stimme selbst verschlüsselt und signiert auf dem Bulletin Board abgibt, kann er erpresst werden, ein bestimmtes Chiffrat abzugeben. Dem kann man aber mit Standardmethoden entgegen gehen. Etwas unschön ist auch, dass der Wähler einen kryptographischen Beweis für die Gültigkeit seiner Stimme durchführen muss.

Kapitel 4

Sicherheitsdefinitionen

4.1 Sicherheitsbegriffe

In den letzten Jahren haben sich verschiedene Begriffe im Bereich der kryptographischen Wahlen etabliert. Auch wenn sich die einzelnen Begriffe nur in Details unterscheiden, kann sich die Bedeutung bzw. die damit verbundenen Garantien um ganze Kassen von Angriffen unterscheiden. Im Folgenden wird die Semantik der wichtigsten Begriffe erläutert. In Abschnitt 4.2 werden dann verschidene formale Definitionen für einige der Begriffe angegeben.

4.1.1 Wahlgeheimnis

Voter-Privacy: (Wahlgeheimnis)
Niemand außer dem Wähler lernt die Stimme, die der Wähler abgibt.

Receipt-Freeness: (Quittungsfreiheit)
Der Wähler kann dem Angreifer, selbst wenn er das wollte, nicht beweisen, für welchen Kandidaten er gestimmt hat.

Coercion-Resistance: (nicht-Erpressbarkeit)
Der Wähler kann vom Angreifer nicht gezwungen werden für einen anderen Kandidaten zu stimmen, als er das eigentlich wollte.

Im Allgemeinen folgt aus der nicht-Erpressbarkeit die Quittungsfreiheit. Aus der Quittungsfreiheit leitet sich das Wahlgeheimnis ab.

4.1.2 Verifizierbarkeit

Individuelle Verifizierbarkeit:
Jeder Wähler kann prüfen, ob seine Stimme korrekt in die Berechnung des Wahlergebnisses eingeflossen ist.

Universelle Verifizierbarkeit:
Jeder kann nachprüfen, ob das Wahlergebnis korrekt berechnet wurde.

Wegen des Wahlgeheimnisses kann natürlicherweise ein Wähler seine eigene Stimmabgabe nur selbst überprüfen. Jedoch sollte jeder nachvollziehen können, dass die abgegebenen und ausreichend individuell überprüften Stimmen tatsächlich zum richtigen Wahlergebnis verarbeitet wurden. Umgekehrt, selbst wenn die Berechnung des Wahlergebnisses anhand der abgegebenen Stimmen von jedem nachvollzogen werden kann, so heißt das noch lange nicht, dass von jedem Wähler die richtige Stimme gezählt wird. Nur, wenn die individuelle Verifizierbarkeit zusätzlich gegeben ist, und man davon ausgehen kann, dass genügend Wähler ihre eigene Stimme überprüfen, kann man auch davon ausgehen, dass das Wahlergebnis tatsächlich korrekt berechnet wurde. Eine der Eigenschaften allein reicht also nicht aus. Nur wenn die individuelle und die universelle Verifizierbarkeit beide gegeben sind, ist ein Wahlverfahren wirklich vollständig verifizierbar. Man spricht in diesem Fall von *Ende-zu-Ende-Verifizierbarkeit*.

Für die vollständige Verfizierbarkeit sind noch weitere Begriffe wichtig, diese sind aber in den meisten Modellen nicht formal definiert:

Berechtigung prüfen:
Nur Stimmen berechtigter Wähler gehen in das Ergebnis ein.

Gleichheit:
Von jedem Wähler kann nur eine Stimme in das Wahlergebnis eingehen.

4.2 Formale Sicherheitsmodelle für kryptographische Wahlverfahren

4.2.1 Modell von Juels, Catalano und Jacobsson (JCJ-Modell)

Juels, Catalano und Jacobsson formulieren in ihrem Papier [JCJ05] mit "Coercion-Resistance" zum ersten Mal eine starke Definition des Begriffs für die Nicht-Erpressbarkeit. Davor war die Nicht-Erpressbarkeit mit der Quittungsfreiheit gleichgesetzt. Juels et. al entdeckten jedoch, dass die Quittungsfreiheit nicht ausreicht, da der Angreifer den Wähler auch erpressen kann, ohne dessen Stimme zu lernen. Leider ist das Modell relativ unflexibel und schwer anwendbar auf andere Wahlverfahren als das in [JCJ05] vorgestellte. Es liefert aber mit seinen Definitionen eine wichtige Grundlage für allgemeiner gehaltene Modelle. Der Vollständigkeit halber definieren Juels et. al. auch die verifizierbare Korrektheit. In dieser Vorlesung wird nur die Definition der Nicht-Erpressbarkeit behandelt, was den interessierten Leser jedoch nicht davon abhalten soll, sich die weiteren Definitionen des Papiers anzuschauen, sie sind ähnlich aufgebaut und auch interessant.

Write-In-Kandidaten werden im Modell nicht berücksichtigt, denn Juels et. al. wollen mit ihrer Definition auch die erzwungene Enthaltung berücksichtigen. Dadurch, dass man mit Write-In-Kandidaten den Wähler immer zwingen kann, für einen sehr unwahrscheinlichen Kandidaten, z.B. einen Zufallsstring, zu stimmen, kann man jedoch effektiv eine Art Enthaltung erzwingen. Write-In-Kandidaten stellen außerdem eine besondere Herausforderung dar, da es dem Wähler möglich ist, seinen Stimmzettel individueller zu *markieren* als es üblicherweise mit einem Kreuz möglich ist. Da auch bei Präsenzwahlen eine erzwungene Enthaltung immer möglich ist, ist das Modell wohl eher für Internetwahlen gedacht.

4.2.1.1 Idee des Papiers

Die Autoren des Papiers sehen in der bisherigen Definition (Receipt-Freeness) verschiedene Schwächen, die den Wähler wieder erpresspar machen.

Randomisation Attack: Bei dieser Attacke hat der Angreifer zwar keine Kontrolle über die Stimme, die der Wähler abgibt, kann aber verhindern, dass der Wähler die Stimme abgeben kann, die er gerne abgeben möchte. Das bedeutet nicht zwangsläufig, dass der Angreifer die Stimme lernt, für die gestimmt wurde. Ein Beispiel wäre hier, den Wähler im Punchscan-Verfahren zu zwingen, das linke Feld zu stempeln. Der Angreifer lernt hier nicht die Stimme, und kann den Wähler auch nicht zwingen, für einen bestimmten Kandidaten zu stimmen, er erreicht aber, dass der Wähler seine Auswahl nicht frei treffen kann.

Forced Abstention Attack: Wie bei der Präsenzwahl kann ein Angreifer einen Wähler erpressen nicht zur Wahl zu gehen, wenn er beispielsweise durch eine Veröffentlichung der Wahlteilnehmer in Erfahrung bringen kann, ob der Wähler zur Wahl gegangen ist. Wie gesagt bieten auch Write-In-Kandidaten eine Möglichkeit für eine erzwungene Enthaltung, sofern das Wahlergebnis genau und vollständig veröffentlicht wird.

Simulation Attack: Kann der Angreifer das Schlüsselmaterial des Wählers in Erfahrung bringen, so ist er in der Lage eine Stimme an Stelle des Wählers abzugeben. Dies ist immer dann ein Problem, wenn der Angreifer echtes Schlüsselmaterial von “unechtem” unterscheiden kann. Den Simulationsangriff verhindert man normalerweise, indem man dem Angreifer unbemerkt ungültige Schlüssel unterjubelt. Dazu darf der Wähler dem Angreifer natürlich nicht die Gültigkeit des echten Schlüssels beweisen können.

4.2.1.2 Bestandteile einer Wahl

Teilnehmer: Die folgenden Gruppen sind an einer Wahl beteiligt:

1. Registrare (*Registrars*): $\mathcal{R} = \{R_1, R_2, \ldots, R_{n_R}\}$
   Die n_R Mitglieder in $\mathcal{R}$ erzeugen gemeinsam das Schlüsselmaterial für den Wähler.
2. Auszähler (*Authorities / Talliers*): $\mathcal{T} = \{T_1, T_2, \ldots, T_{n_R}\}$
   Sie verarbeiten die Stimmzettel und werten sie gemeinsam aus. Zum Schluss veröffentlichen sie das Ergebnis.

3. Wähler (*Voters*): $\mathcal{V} = \{V_1, V_2, \ldots, V_{n_V}\}$
   $\mathcal{V}$ ist die Menge der Wähler. Sie wird durch $\mathcal{R}$ administriert. Im folgenden wird i als Identifikator für einen Wähler verwendet.

Bausteine: Die folgenden Bausteine werden für die Wahl benötigt:

1. Schwarzes Brett (*Bulletin Board*):
   Die Funktion des Bulletin Boards ($\mathcal{BB}$) ist die gewohnte: Jeder kann es lesen, und Wahlteilnehmer können Nachrichten anhängen.
2. Kandidatenliste (*candidate slate*): $\vec{\mathcal{C}} = \{C_1, C_2, \ldots, C_{n_C}\}$
   $\vec{\mathcal{C}}$ ist eine geordnete Liste von n_C Identifikatoren, die jeweils einer Wahlmöglichkeit (z.B. Kandidat oder Partei) fest zugeordnet sind. Wir nehmen zur Vereinfachung an, dass die Kandidaten durch Zahlen von 1 bis n_C repräsentiert werden. Damit ist die Kandidatenliste durch n_C eindeutig bestimmt.
3. Auszählung (*tally*): $\vec{\mathcal{X}} = \{X_1, X_2, \ldots, X_{n_C}\}$
   Der Vektor $\vec{\mathcal{X}}$ repräsentiert das Ergebnis der Wahl. Dabei steht der j-te Eintrag für die Zahl der für den j-ten Kandidaten abgegebenen Stimmen.

Funktionen: Eine Wahl besteht aus den folgenden Funktionen:

1. registrieren (*register*): $register(SK_R, i, k_1) \rightarrow (sk_i, pk_i)$
   Die Funktion nimmt den geheimen Schlüssel SK_R der Registrare, einen Identifikator i für einen Wähler und einen Sicherheitsparameter k_1 entgegen. Sie gibt ein Schlüsselpaar (sk_i, pk_i) für den Wähler V_i aus. Die Ausgabe wird gemeinsam von den Teilnehmern der Gruppe $\mathcal{R}$ und ggf. dem Wähler V_i berechnet.
2. wählen (*vote*): $vote(sk, PK_{\mathcal{T}}, n_C, \beta, k_2) \rightarrow ballot$
   Die Funktion nimmt einen geheimen Schlüssel (des Wählers) sk, den öffentlichen Schlüssel $PK_{\mathcal{T}}$ der Auszähler, die Kandidatenliste, repräsentiert durch den höchsten Identifikator n_C, die Wahl des Wählers β und einen Sicherheitsparameter k_2 entgegen und gibt einen Stimmzettel aus.
3. auszählen (*tally*): $tally(SK_{\mathcal{T}}, \mathcal{BB}, n_C, \{pk_i\}_{i=1}^{n_V}, k_3) \rightarrow (\vec{X}, P)$
   Die Auszählfunktion erhält den geheimen Schlüssel $SK_{\mathcal{T}}$ der Auszählenden, den gesamten Inhalt des Bulletin Boards, die Kandidatenliste n_C, die öffentlichen Schlüssel $\{pk_i\}_{i=1}^{n_V}$ aller Wähler und einen Sicherheitsparameter k_3. Ausgegeben wird eine Ergebnisliste $\vec{X}$ und ein interaktiver Beweis P, mit dessen Hilfe kontrolliert werden kann, dass das Ergebnis korrekt berechnet wurde.
4. überprüfen (*verify*): $verify(PK_{\mathcal{T}}, \mathcal{BB}, n_C, \vec{X}, P) \rightarrow \{0, 1\}$
   Die Funktion *verify* überprüft, ob der Beweis P korrekt ist. Dazu erhält sie den öffentlichen Schlüssel $PK_{\mathcal{T}}$ der Auszählenden, den Inhalt des Bulletin Boards, die Kandidatenliste, das Auszählungsergebnis und den zu prüfenden Beweis. Als Ergebnis gibt die Funktion "1" aus, wenn der Beweis korrekt ist oder "0" in allen anderen Fällen.

Ein Wahlverfahren besteht dann aus den vier Funktionen $ES = (register, vote, tally, verify)$.

4.2.1.3 Wahlphasen und Angreifermodell

Im folgenden wird dargestellt, welche Phasen eine Wahl durchläuft. In *kursivschrift* wird dabei angedeutet, an welchen Stellen der Angreifer wen wie korrumpieren kann.

Setup Das Schlüsselmaterial für $\mathcal{R}$ und $\mathcal{T}$ wird generiert. Die Kandidatenliste $\mathcal{C}$ wird von $\mathcal{R}$ mit angemessenem Integritätsschutz veröffentlicht.
Während der Setup-Phase darf der Angreifer eine Minderheit der Teilnehmer in $\mathcal{R}$ und $\mathcal{T}$ korrumpieren, und zwar statisch und aktiv. Statisch heißt, korrumpierte Teilnehmer bleiben korrumpiert, der Angreifer kann nicht einen Teilnehmer "freilassen" und dafür einen anderen korrumpieren. Aktiv heißt, der Angreifer kontrolliert die korrumpierten Teilnehmer vollständig.

Registrierung *Vor der Registrierung darf der Angreifer den Wähler erpressen, ihm später ein Transkript des Registrierungsvorgangs zu geben. Er darf ihm außerdem in die Registrierung Daten mitgeben, die der Wähler zu verwenden hat, wie z.B. Zufallswerte für die Generierung der*

Wählerschlüssel. Während der Registrierung selbst findet keine Korruption statt. Erpresste Wähler werden während der Registrierung nicht vom Angreifer beobachtet und können somit z.B. über das Transkript lügen.
Die Idenditäten der Teilnehmer werden von $\mathcal{R}$ geprüft. Nach erfolgreicher Prüfung wird ein Teilnehmer zum *registrierten Wähler* und bekommt von $\mathcal{R}$ eine Wahlerlaubnis (engl. *Credential*) für die Teilnahme an der Wahl. Diese dürfen ggf. in weiteren Wahlen wiederverwendet werden. $\mathcal{R}$ veröffentlicht eine Wählerliste (engl. *voter roll*) L, in der die Wähler stehen, die wahlberechtigt sind.

Wärend der folgenden 3 Phasen darf der Angreifer eine Minderheit in $\mathcal{T}$ und eine beliebige Anzahl Wähler statisch und aktiv korrumpieren. Er kann nun auch versuchen, einige Wähler zu erpressen. Diese Wähler kontrolliert er nicht vollständig, aber kann mit ihnen kommunizieren. Er kann z.B. von ihnen verlangen, Schlüsselmaterial herauszugeben, über das die Wähler wiederum evtl. lügen können, u.s.w. In der weiteren Beschreibung des Modells gehen wir der Einfachheit halber davon aus, dass der Angreifer versucht, genau einen Wähler zu erpressen. Das Modell kann aber leicht auf mehrere erpresste Wähler verallgemeinert werden, siehe dazu [JCJ05]

Wählen Registrierte Wähler nutzen ihr *Credential*, um bzgl. der Kandidatenliste $\mathcal{C}$ zu wählen. *Der Wähler hat dabei einen anonymen Kanal, d.h. der Angreifer sieht nicht, welcher Stimmzettel von welchem Wähler kommt.*

Auszählen Die Auszähler in $\mathcal{T}$ verarbeiten den Inhalt des $\mathcal{BB}$, berechnen den Auszählungs-Vektor $\mathcal{X}$ und einen Korrektheitdsbeweis P.

Verifizieren Jeder (Auch nicht-Teilnehmer) kann den Inhalt von $\mathcal{BB}$, den Beweis P und die Wählerliste L benutzen und $verify$ ausführen, um die Korrektheit der Auszählung $\mathcal{X}$ zu prüfen.

4.2.1.4 Definition von Nicht-Erpressbarkeit

Die Nicht-Erpressbarkeit wird hier, wie es in der Kryptographie heutzutage gängig ist, über ein Angriffsspiel definiert: Der Angreifer findet sich in einem von zwei Experimenten wieder: dem "realen" Experiment $exp_{ES,\mathcal{A}}^{c-resist}$, in dem jeweils mit Wahrscheinlichkeit $\frac{1}{2}$ der erpresste Wähler entweder dem Angreifer gehorcht und ihm sein geheimes credential gibt, oder ihm einen falschen Schlüssel unterjubelt und selbst seinen Wunschkandidaten wählt, und dem idealen Experiment $exp_{ES,\mathcal{A}}^{c-resist-Ideal}$, in dem der Angreifer mit 50%iger Wahrscheinlichkeit eine Stimme statt dem erpressten Wähler abgeben kann, aber außer dem Wahlergebnis keine Information lernt. Die nicht-Erpressbarkeit wird dann definiert über die *Ununterscheidbarkeit* der beiden Experimente. Kann der Angreifer für ein Wahlverfahren nicht nennenswert besser als raten, ob der Wähler sich erpressen lässt, als in einem idealen Wahlverfahren, so erfüllt dieses Wahlverfahren die Nicht-Erpressbarkeit. Vor der formalen Definition der Nicht-Erpressbarkeit brauchen wir nun noch ein paar Notationen:

FakeKey Um die nicht-Erpressbarkeit formal definieren zu können, geben Juels et. al dem Wahlverfahren eine weitere Funktion

$$fakeKey(PK_{\mathcal{T}}, sk, pk,) \rightarrow \tilde{sk}.$$

Diese Funktion bekommt den Public Key der Auszähler und das (echte) Schlüsselpaar (*credential*) des Wählers als Eingabe, und generiert daraus einen *ungültigen* Schlüssel $\tilde{sk}$, der für jeden außer einer Mehrheit in $\mathcal{T}$ ununterscheidbar von einem gültigen Schlüssel ist.

Enthaltung und ungültige Stimmzettel Unter den Stimmzetteln wird eine Enthaltung (*null ballot*) mit Φ bezeichnet, und ein mit ungültigem Schlüssel abgegebener Stimmzettel mit λ.

Vorwissen des Angreifers Bei Wahlverfahren leben wir nicht in einer idealen Welt, in der der Angreifer über die Eingabe nicht korrumpierter Parteien keinerlei Information hat. Der Angreifer hat normalerweise ein unsicheres Wissen darüber, wie sich nicht korrumpierter Wähler verhalten werden. Man kann davon ausgehen, dass er weiß, welche Kandidaten eher viele Stimmen bekommen werden, und welche eher wenige oder vielleicht garkeine. Er weiß vielleicht auch, was bestimmte

Wähler wahrscheinlich wählen wollen, weil er ihre politischen Ansichten kennt. Dieses unsichere Vorwissen des Angreifers wird modelliert durch eine dem Angreifer bekannte Wahrscheinlichkeitsverteilung D_{n,n_C} über den Vektoren $(\beta_1, \ldots, \beta_n) \in (\{1, \ldots, n_C\} \cup \Phi \cup \lambda)^n$ in der Menge der möglichen Stimmabgaben der n Wähler.

Gesamte Stimmabgabe Die gesamte Stimmabgabe aller Wähler bezüglich der Verteilung D_{n,n_C} notieren wir mit

$$vote(\{sk_i\}, PK_{\mathcal{T}}, n_C, D_{n,n_C}, k_2).$$

Sonstige Notationen Mit n_V bezeichnen wir die Anzahl wahlberechtigter Wähler, mit n_A die Wähler, die vollständig vom Angreifer kontrolliert werden. Damit ist $n_u := n_V - n_A - 1$ die Anzahl aus Sicht des Angreifers unsicherer Stimmabgaben (also die Anzahl der unkorrumpierten Wähler), wenn der Angreifer versucht, genau einen Wähler zu erpressen.

Im Angriffsspiel bezeichen wir mit $:=$ eine Zuweisung und mit $\Leftarrow$ die *append*-Funktion des Bulletin Boards, also das Anhängen von Daten. Mit $x \in_R \mathcal{M}$ bezeichnen wir die zufällige, gleichverteilte Auswahl eines Elements x aus einer Menge $\mathcal{M}$. Mit $Z := \mathcal{A}(x, \text{“}y''\text{)}$; bezeichnen wir die Ausgabe Z eines Angreifers $\mathcal{A}$, der x als Eingabe bekommt, bzw. dem x bekannt ist, und der "y" in diesem Schritt zum Ziel hat.

Das Experiment $exp^{c-resist}_{ES,\mathcal{A}}$ Hier nun der erste Teil des Angriffsspiels: das Experiment, in dem der erpresste Wähler mit 50%iger Wahrscheinlichkeit dem Angreifer gehorcht:

Experiment $exp^{c-resist}_{ES,\mathcal{A}}(k_1, k_2, k_3, n_V, n_A, n_C)$

$V := \mathcal{A}(voter\,names, \text{“}control\,voters''\text{)};$	$\mathcal{A}$ korrumpiert Wähler in V vor Registrierungsphase
$\{(sk_i, pk_i) := register(SK_{\mathcal{R}}, i, k_2)\}_{i=1}^{n_V};$	Registrierung von i für alle Wähler $i = 1, \ldots, n_V$
$(j, \beta) := \mathcal{A}(\{sk_i\}_{i \in V}, \text{“}set\,target\,voter''\text{)};$	$\mathcal{A}$ versucht, Wähler j zu erpressen
$if((\lvert V\rvert \neq n_A)\, or\, (j \notin \{1, \ldots, n_V\} \setminus V)$	
$or\, \beta \notin \{1, \ldots, n_C\} \cup \Phi)\, then$	Angreifer-Ausgaben werden auf Gültigkeit geprüft,
$\quad output\, 0;$	sind sie ungültig, wird sofort mit Ausgabe 0 beendet.
$b \in_R \{0, 1\};$	Ein Zufallsbit b wird gewürfelt
$if\, b = 0\, then$	Falls $b = 0$, lässt sich Wähler j nicht erpressen,
$\quad \tilde{sk} := fakeKey(PK_{\mathcal{T}}, sk_j, pk_j);$	sondern gibt dem Angreifer einen falschen Schlüssel,
$\quad \mathcal{BB} \Leftarrow vote(sk_j, PK_{\mathcal{T}}, n_C, \beta, k_2);$	und wählt, was er selbst wählen will.
$else$	Falls $b = 1$ läßt sich der Wähler j erpressen,
$\quad \tilde{sk} = sk_j;$	und gibt dem Angreifer seinen secret key
$\mathcal{BB} \Leftarrow vote(\{sk_i\}_{i \neq j, j \notin V}, PK_{\mathcal{T}}, n_C, D_{n_u,n_C}, k_2);$	die ehrlichen Wähler wählen
$\mathcal{BB} \Leftarrow \mathcal{A}(\tilde{sk}, \mathcal{BB}, \text{“}cast\,ballots''\text{)};$	$\mathcal{A}$ wählt mit Schlüssel $\tilde{sk}$
$(\mathcal{X}, P) := tally(SK_{\mathcal{T}}, \mathcal{BB}, n_C, \{pk_i\}_{i=1}^{n_V}, k_3);$	Die Stimmen werden ausgezählt
$b' := \mathcal{A}(\mathcal{X}, P);$	Angreifer “rät" das Bit b
$if\, b' = b\, then$	
$\quad output\, 1;$	
$else$	
$\quad output\, 0;$	

Dieses Experiment wird nun mit einem idealen Experiment verglichen, in dem die Auszählungsfunktion *tally* durch die Funktion *idealTally* ersetzt wird, die nur die Stimmen der ehrlichen Wähler berücksichtigt, und doppelte Stimmabgaben verwirft.

Das ideale Experiment $exp^{c-resist-ideal}_{ES,\mathcal{A}}$ Das ideale Experiment wird durchgeführt mit einem zweiten, relativ dummen Angreifer $\mathcal{A}'$, der zwar die secret keys korrumpierter Wähler bekommt, diese ihm aber nichts nützen, da hiermit abgegebene Stimmen von der *idealTally* nicht berücksichtigt werden. Auch in diesem idealen Experiment muss der echte Angreifer $\mathcal{A}$ das bit b raten (er weiß nicht, ob er sich in diesem Experiment befindet!). Im zweiten Experiment bekommt nur der Dumme Angreifer $\mathcal{A}'$ den secret key des Wählers, der hieraus nichts lernt. Da $ideal - tally$ doppelte Stimmabgaben verwirft, und der Wähler auch hier mit 50 %iger Wahrscheinlichkeit selbst eine Stimme abgibt, wird die Angreiferstimme mit 50 %iger Wahrscheinlichkeit nicht gezählt. Der Angreifer $\mathcal{A}$ bekommt hier nun als einzige Entscheidungsgrundlage die Auszählung.

Experiment $exp^{c-resist-Ideal}_{ES,\mathcal{A}}(k_1, k_2, k_3, n_V, n_A, n_C)$

$V := \mathcal{A}'(voter\,names, \text{“}control\,voters''); $	$\mathcal{A}'$ korrumpiert Wähler in V vor Registrierungsphase
$\{(sk_i, pk_i) := register(SK_{\mathcal{R}}, i, k_2)\}_{i=1}^{n_V};$	Registrierung von i für alle Wähler $i = 1, \ldots, n_V$
$(j, \beta) := \mathcal{A}'(\{sk_i\}_{i \in V}, \text{“}set\,target\,voter'');$	$\mathcal{A}'$ versucht, Wähler j zu erpressen
$if((\lvert V\rvert \neq n_A)\,or\,(j \notin \{1, \ldots, n_V\} \setminus V)$ $or\,\beta \notin \{1, \ldots, n_C\} \cup \Phi)\,then$	Angreifer-Ausgaben werden auf Gültigkeit geprüft, sind sie ungültig, wird sofort mit Ausgabe 0 beendet.
$\quad output\,0;$	
$b \in_R \{0, 1\};$	Ein Zufallsbit b wird gewürfelt
$if\,b = 0\,then$	Falls $b = 0$, lässt sich Wähler j nicht erpressen,
$\quad \mathcal{BB} \Leftarrow vote(sk_j, PK_{\mathcal{T}}, n_C, D_{n_U, n_C}, k_2)$	sondern wählt, was er will
$\tilde{sk} \Leftarrow sk_j;$	$\mathcal{A}'$ bekommt sk_j, lernt daraus nichts
$\mathcal{BB} \Leftarrow vote(\{sk_i\}_{i \neq j, j \notin V}, PK_{\mathcal{T}}, n_C, D_{n_u, n_C}, k_2);$	die ehrlichen Wähler wählen
$\mathcal{BB} \Leftarrow \mathcal{A}'(\tilde{sk}, \mathcal{BB}, \text{“}cast\,ballots'');$	$\mathcal{A}'$ wählt mit $\tilde{sk}$, $idealTally$ verwirft Stimme bei $b = 0$
$(\mathcal{X}, P) := idealTally(SK_{\mathcal{T}}, \mathcal{BB}, n_C, \{pk_i\}_{i=1}^{n_V}, k_3);$	Die Stimmen werden ausgezählt
$b' := \mathcal{A}(\mathcal{X}, P);$	Angreifer “rät" das Bit b
$if\,b' = b\,then$	
$\quad output\,1;$	
$else$	
$\quad output\,0;$	

Ein Wahlverfahren ist dann nicht-erpressbar, wenn der Angreifer im realen Experiment b gleich gut rät wie im idealen Experiment, man von außen die Experimente also nicht *unterscheiden* kann. Ist dies der Fall, so ist das reale Experiment genauso nicht-Erpressbar wie das ideale Experiment, in dem der Angreifer als einzige Information die Auszählung bekommt. (Die secret keys der korrumpierten Wähler bekommt nur der dumme Angreifer $\mathcal{A}'$, der aber daraus nichts lernt und auch nie das $\mathcal{BB}$ sieht.)

Definition 1 (nicht-Erpressbarkeit) *Ein Wahlverfahren $ES = (register, vote, tally, verify)$ mit Sicherheitsparametern k_1, k_2, k_3 für die entsprechenden Funktionen $register, vote$ und $tally$ ist bei n_V wahlberechtigten Wählern, n_A korrumpierten Wählern und n_V Kandidaten* nicht-Erpressbar, *wenn gilt*

$$|Pr[exp_{ES,\mathcal{A}}^{c-resist}(k_1, k_2, k_3, n_V, n_A, n_C) \to 1] - Pr[exp_{ES,\mathcal{A}}^{c-resist-Ideal}(k_1, k_2, k_3, n_V, n_A, n_C) \to 1]| \leq \epsilon,$$

wobei ϵ eine in den Sicherheitsparametern vernachlässigbare Funktion ist.

Wie man sieht, wird die Ununterscheidbarkeit der Experimente dadurch modelliert, dass der Angreifer in beiden Experimenten gleichwahrscheinlich das Bit b richtig rät, also beide Experimente gleichwahrscheinlich eine 0 oder eine 1 ausgeben.

4.2.1.5 Vor- und Nachteile des Modells

Das JCJ-Modell ist eins der ersten Modelle, das eine starke Definition von Nicht-Erpressbarkeit liefert, und die Nicht-Erpressbarkeit klar von der Quittungsfreiheit unterscheidet. Ein Nachteil des Modells ist, dass die Definitionen nicht sehr allgemein gehalten sind, und es deshalb schwer bis unmöglich ist, weitere Wahlverfahren in dem Modell zu untersuchen. Außerdem werden hier write-In-Kandidaten und Präsenzwahlen vom Modell ausgesschlossen, da hier eine erzwungene Enthaltung immer möglich ist.

4.2.2 Das Modell von Küsters, Truderung und Vogt

Das Modell von Küsters, Truderung und Vogt [KTV12] ist deutlich flexibler als das JCJ-Modell, es ist allgemeiner gehalten und kann daher besser auf unterschiedliche Wahlverfahren angewendet werden. In diesem Modell werden Nicht-Erpressbarkeit und Verifizierbarkeit definiert, allerdings in zwei verschiedenen Veröffentlichungen. In dieser Vorlesung wird nur die Definition der Nicht-Erpressbarkeit behandelt. Auch wenn dieses Modell und das JCJ-Modell auf den ersten Blick völlig unterschiedlich aussehen, finden sich doch beim genaueren Hinsehen starke Parallelen. Die Idee der Definition von Nicht-Erpressbarkeit ist, dass der Angereifer nicht unterscheiden können soll, ob der Wähler den Anweisungen des Angreifers gehorcht, oder ob er eine *Ausweichstrategie* ausführt. Das Modell grenzt sich dadurch von anderen Modellen ab, dass es nicht direkt festlegt, ob ein Verfahren die Nicht-Erpressbarkeit erfüllt oder nicht, sondern es misst, in welchem Maße δ ein Verfahren die Nicht-Erpressbarkeit erfüllt. So ist es diesem Modell z.B. möglich, Write-In-Kandidaten zu betrachten. Eine Wahl verliert dadurch nicht wie im JCJ-Modell die nicht-Erpressbarkeit, sondern

das Erpressbarkeitsmaß δ erhöht sich. Zum Vergleich gibt das Originalpapier [KTV12] ein ideales Wahlverfahren an, anhanddessen das kleinste zu erreichende δ berechnet wird. Weitere Wahlprotokolle können damit auf das ideale Protokoll reduziert werden, um die Optimalität des Verfahrens bzgl. δ zu beweisen. Da das Modell die Teilnehmer als probabilistische, polynomial beschränkte Turingmaschienen modelliert, betrachtet es ausschließlich polynomiell beschränkte Angreifer.

4.2.2.1 Modell eines Wahlverfahrens

Ein Wahlverfahren P spezifiziert, wie sich ehrliche Teilnehmer verhalten, und impliziert ein Wahlsystem $ES = P(k, m, n, \vec{p})$, wobei k die Anzahl Kandidaten bzw. Auswahlmöglichkeiten (ohne Enthaltung) ist, m die Anzahl Wähler, n die Anzahl ehrlicher Wähler ($n \leq m$) und $\vec{p} = p_0, \ldots, p_k$ die Wahrscheinlichkeitsverteilung der möglichen Auswahlmöglichkeiten, dabei ist p_0 die Wahrscheinlichkeit für eine Enthaltung. Die k Auswahlmöglichkeiten sind nicht weiter spezifiziert, es kann sich um Kandidaten handeln, aber auch um ein Ranking oder die verschidenen Arten, einen Stimmzettel auszufüllen (z.B. bei ThreeBallot). Die Anzahl n der ehrlichen Wähler kann auch gesehen werden als die Mindestanzahl ehrlicher Wähler, die das Protokoll verlangt. Die $m - n$ unehrlichen Wähler können beliebig vom Protokoll abweichen. Den Parameter $\vec{p}$ finden wir im JCJ-Modell als die Wahrscheinlichkeitsverteilung D_{n,n_C} wieder. Er modelliert genauso die Tatsache, dass der Angreifer ein unsicheres Vorwissen darüber hat, wie gewählt wird. Der Angreifer weiß zum Beispiel, welche Parteien beliebter sind als andere, welche Kandidaten wahrscheinlich nur wenige oder keine Stimmen bekommen, oder etwa welche Ankreuzmuster bei einer Rangfolgewahl eher selten sind.

4.2.2.2 Notationen

Die Protokollteilnehmer sind interaktive Turingmaschenen (ITM), die über Ein- und Ausgabebänder kommunizieren und PPT-Algorithmen ausführen, also probabilistische Algorithmen mit (im Sicherheitsparameter und der Eingabelänge) polynomieller Laufzeit.

Vernachlässigbare Funktion (negligible function) Eine Funktion $f : \mathbb{N} \to \mathbb{R}$ ist *vernachlässigbar*, wenn für jedes $c > 0$ ein $l_0 \in \mathbb{N}$ existiert, so dass

$$f(l) \leq \frac{1}{l^c}$$

für alle $l > l_0$ gilt.

δ-begrenzt (δ-bounded) Eine Funktion $f : \mathbb{N} \to \mathbb{R}$ ist *δ-bounded*, wenn für jedes $c > 0$ ein $l_0 \in \mathbb{N}$ existiert, so dass

$$f(l) \leq \delta + \frac{1}{l^c}$$

für alle $l > l_0$ gilt. Die Funktion f ist also begrenzt durch eine Schranke δ plus eine vernachlässigbare Funktion.

ITM-System Ein ITM-System S ist eine Multimenge von ITMs

$$S = M_1 || \ldots || M_l,$$

wobei $M_1, \ldots, M_l$ ITMs oder ITM-Systeme sind. In einem ITM-System S existiert höchstens eine ITM mit speziellem Ausgabeband *desicion*. Für einen Sicherheitsparameter l ist dann

$$Pr[S^{(l)} \mapsto 1]$$

die Wahrscheinlichkeit, dass S auf dem Band *desicion* eine 1 ausgibt.

Lauf (Run) Ein *Lauf* einer ITM oder eines ITM-Systems S ist wie der Name schon sagt ein kompletter Durchlauf oder ein Ausführen der ITM oder des ITM-Systems. Ein Lauf ist eindeutig bestimmt durch den Zufall, den die ITM bzw. die ITMs in S in diesem Lauf benutzen. In einem Lauf eines ITM-Systems werden ein bis mehrere ITMs des ITM-Systems nacheinander oder parallel ausgeführt, es müssen also nicht notwendigerweise alle ITMs im System ausgeführt werden. Implizit wird ein Scheduler angenommen, der dafür sorgt, dass die richtigen ITMs zur richtigen Zeit drankommen.

Eigenschaft (Property) Eine *Eigenschaft* γ eines ITM-Systems S ist eine Untermenge der möglichen Läufe von S. Für eine Eigenschaft γ von S bezeichnen wir mit

$$Pr[S^{(l)} \mapsto \gamma]$$

die Wahrscheinlichkeit, dass ein Lauf von S zu γ gehört (mit Sicherheitsparameter l). Ein Beispiel für eine solche Eigenschaft γ wäre "alle Läufe, in denen der Wähler für Kandidat i stimmt und die Stimme auch korrekt gezählt wird ".

4.2.2.3 Modellierung der Protokollteilnehmer

Ein Wahlsystem $ES = P(k, m, n, \vec{p})$ spezifiziert PPT-ITMs für die ehrlichen Teilnehmer. Weiterhin werden der Angreifer und der Wähler als PPT-ITM-Systeme modelliert.

Ehrliche Voting Authorities $S_1, \ldots, S_l$: Die ehrlichen Voting authorities bestehen aus den ITM-Systemen $S_1, \ldots, S_l$. Sie halten sich ans durch P spezifizierte Protokoll.

Ehrliche Wähler $S_{V_1}, \ldots, S_{v_n}$: Die ehrlichen Wähler sind modelliert durch die ITM-Systeme $S_{V_1}, \ldots, S_{v_n}$. Sie halten sich ans Protokoll und treffen ihre Wahl gemäß der Verteilung $\vec{p}$, falls sie sich nicht enthalten. Falls sie sich enthalten, führen sie das Wahlprotokoll evtl. nicht aus.

Angreifer (Erpresser) C_S: Der Angreifer ist ein ITM-System C_S. Dieses System enthält *alle* PPT-Algorithmen, und damit alle möglichen Erpress-Strategien. Die einzige Einschränkung ist die Schnittstelle zum Rest des Systems, die muss natürlich passen. Die ITMs in C_S können sich mit allen unehrlichen Teilnehmern (Wähler und Authorities) direkt verbinden. Außerdem können sie mit dem erpressten Wähler kommunizieren, alle öffentliche Information lesen und Teile des Netzwerks abhören.

Erpresster Wähler V_S: Der erpresste Wähler ist ein ITM-System V_S, das ebenfalls alle PPT-ITMs enthält, insbesondere kann er alles tun, worum der Angreifer ihn bitten könnte, er enthält aber auch alle Ausweichstrategien. Der erpresste Wähler enthält seine Schnittstelle zum Wahlprotokoll als ehrlicher Wähler sowie eine Schnittstelle zur Kommunikation mit dem Angreifer. Weiterhin enthält V_S eine *Dummy-Strategie dum*, diese leitet einfach alle Nachrichten zwischen der Schnittstelle mit dem Angreifer und der Schnittstelle als ehrlicher Wähler weiter, d.h. der Wähler gibt unverändert alle Nachrichten vom Angreifer ans Wahlprotokoll und alle Protokollantworten an den Angreifer weiter.

ITM-System es aller ehrlichen Teilnehmer: Die ehrlichen Teilnehmer ergeben zusammen das ITM-System $es = (S_{V_1}|| \ldots ||S_{v_n}||S_1|| \ldots ||S_l)$.

Instanz von es: Ein ITM-System $(c||v||es)$ ist eine *Instanz* von es, wobei $c \in C_S$ die Erpress-Strategie des Angreifers und $v \in V_S$ die vom Wähler ausgeführte ITM ist.

4.2.2.4 Definition der Nicht-Erpressbarkeit

Die Idee der Definition ist folgende: Der Wähler hat als Ziel eine Eigenschaft γ. Falls der Wähler den Kandidaten i wählen möchte, wäre ein sinnvolles γ somit sie Menge der Läufe, in denen der Wähler für i stimmt, und die Stimme korrekt gezählt wird. Das Ziel des Angreifers ist, dass der Wähler die Dummystrategie *dum* ausführt, denn somit hätte der Angreifer die volle Kontrolle über den Wähler. Die Nicht-Erpressbarkeit ist nun dann gegeben, wenn es für den Wähler eine Ausweichstrategie $\tilde{v_S}$ gibt, mit der der Wähler sein Ziel γ erreicht, der Angreifer aber nicht unterscheiden kann, ob der Wähler *dum* oder $\tilde{v_S}$ ausführt. Diese Möglichkeit des Angreifers, zwischen $\tilde{v_S}$ und *dum* zu unterscheiden, wird nun gemessen in einem Parameter δ. Genau hier liegt der Unterschied zwischem diesem Modell und den meisten anderen: Es wird nicht gesagt, ob ein Verfahren die nicht-Erpressbarkeit erfüllt, sondern inwiefern, also in welchem Maß δ.

Definition 2 (δ-Nicht-Erpressbarkeit) *Sei P ein Wahlprotokoll, $S = P(k, m, n, \vec{p})$ ein Wahlsystem. Sei $\delta \in [0, 1]$ und γ eine Eigenschaft von S. Das System S ist δ-nicht-Erpressbar bezüglich γ, wenn es eine Ausweichstrategie $\tilde{v_S}$ gibt, so dass für alle Erpress-Strategien $c \in C_S$ gilt:*

1. $Pr[(c||\tilde{v_S}||es)^l \mapsto \gamma]$ *ist überwältigend als Funktion im Sicherheitsparameter* l*, und*
2. $|Pr[(c||dum||es)^l \mapsto 1] - Pr[(c||\tilde{v_S}||es)^l \mapsto 1]|$ *ist* δ*-bounded als Funktion im Sicherheitsparameter* l*.*

Wie man sieht, verlangt die Definition *eine* Ausweichstrategie für *alle* möglichen Angreifer und nicht umgekehrt. Damit gehört zu jedem δ-nicht-erpressbaren Wahlverfahren eine spezielle Ausweichstrategie $\tilde{v_S}$, die nach Meinung der Autoren von [KTV12] auch im Wahlverfahren angegeben sein sollte.

4.2.2.5 Das ideale Protokoll und das minimale δ

Das Erpressbarkeitsmaß δ hängt nicht nur vom Wahlverfahren selbst, sondern auch von äußeren Gegebenheiten ab, wie z.B. der Anzahl Wähler, die zur Wahl gehen (bei zwei Wählern weiß der eine immer, was der andere gewählt hat), oder von der Wahrscheinlichkeitsverteilung der Wählerstimmen (der Angreifer kann den Wähler immer zwingen, einen Kandidaten zu wählen, der sonst wahrscheinlich 0 Stimmen bekommen hätte). Gerade weil der Angreifer die Wahrscheinlichkeitsverteilung der Stimmen kennt und das Wahlergebnis sieht, wird es kein Verfahren geben, bei dem die Differenz δ ganz verschwindet. Um herauszufinden, welches das minimale δ ist, was überhaupt erreicht werden kann, wird in [KTV12] ein ideales Protokoll P_{Ideal} definiert, das alle vom Wahlverfahrenden hinzukommenden Faktoren ausblendet, so dass nur diese äusseren Faktoren übrig bleiben, die unabhängig vom Wahlverfahren sind und damit bei jeder Wahl zum Erhöhen von δ beitragen. Genauer sind diese äußeren Gegebenheiten genau die Parameter k, n, und $\vec{p}$. Das vom idealen Protokoll P_{Ideal} erreichte δ bezeichnen wir mit δ_{min}. Kein Wahlverfahren kann bei gleichen äußeren Gegebenheiten δ-nicht-erpressbar sein für ein $\delta < \delta_{min}$. Für weitere Wahlverfahren kann mit Hilfe von P_{Ideal} bewiesen werden, dass sie optimal bzgl. δ sind, indem sie auf P_{Ideal} reduziert werden.

Nun zur Beschreibung des idealen Wahlverfahrens $S = P_{Ideal}(k, m, n, \vec{p})$.

Teilnehmer: Das Potokoll P_{Ideal} hat folgende Teilnehmer:

- **Wahlauthority:** Das Protokoll hat genau eine Wahlauthority, und die ist ehrlich. Sie nimmt von den Wählern die Stimmen entgegen, berechnet ehrlich und korrekt die Auszählung und gibt sie bekannt. Außer dem Ergebnis der Auszählung hat die Wahlauthority keine Ausgaben. Es gibt also von Seiten der Wahlautghority weder Protokollantworten für die Wähler noch irgendeine öffentliche Information außer dem Wahlergebnis.
- **Ehrliche Wähler:** Jeder Wähler hat einen *untappable channel* (abhörsicherer Kanal) zur Wahlauthority. Ehrliche Wähler wählen einen Kandidaten bezüglich $\vec{p}$ und teilen diesen entweder über den *untappable channel* der Wahlauthority mit oder enthalten sich.
- **Angreifer:** Der Angreifer kennt die äußeren Gegebenheiten k, n und $\vec{p}$. Er sieht seine eigenen *random coins* und das Wahlergebnis, sonst nichts. Die *random coins* bezeichnen den Zufall in einem Lauf einer probabilistischen Turingmaschine. Ein Lauf einer PPT-ITM ist durch diesen Zufall eindeutig bestimmt.
- **Erpresster Wähler:** Der erpresste Wähler kann mit dem Angreifer kommunizieren, hat aber genau wie die ehrlichen Wähler einen untappable channel zur Wahlauthority.

Protokollablauf: Der Protokollablauf ist im Prinzip schon aus der Beschreibung der Teilnehmer extrahierbar, wird hier jedoch nochmal explizit gemacht:

1. Der Wähler trifft seine Wahl bezüglich $\vec{p}$.
2. Der Wähler enthält sich oder teilt seine Wahl über den untappable Channel der Wahlauthority mit.
3. Die Wahlauthority speichert die Wählerstimmen.
4. Die Schritte 1. bis 3. werden für alle ehrlichen Wähler ausgeführt.
5. Die Wahlauthority berechnet nach der Wahlphase ehrlich und korrekt das Wahlergebnis und gibt es ohne weitere Information bekannt.

Gegenstrategie: Angenommen, der erpresste Wähler möchte für den Kandidaten i stimmen, sein Ziel ist also die Eigenschaft γ_i, die alle Läufe enthält, in denen der erpresste Wähler erfolgreich eine Stimme für i abgibt. Er bekommt vom Angreifer entweder die Anweisung, für einen Kandidaten j zu stimmen, oder sich zu enthalten. Die Gegenstrategie sieht nun folgendermaßen aus:

- Falls der Angreifer den Wähler anweist, sich zu enthalten, so enthält er sich.
- Falls er Angreifer den Wähler anweist, für einen Kandidaten j zu stimmen, so stimmt er stattdessen für Kandidat i.

Wegen des untappable channel bekommt der Angreifer es nicht direkt mit, dass der Wähler nicht für j sondern für i gestimmt hat. Seine einzigen Anhaltspunkte sind $\vec{p}$, das Wahlergebnis und die Stimmen der korrumpierten Wähler, die er vom Wahlergebnis abziehen kann.

Berechnung von δ_{min}: Der Angreifer kennt die Stimmen der korrumpierten Wähler, und kann sie vom Wahlergebnis abziehen. Dadurch kann er sich ein sogenanntes *pure result* berechnen, das nur aus den Stimmen der ehrlichen Wähler und des erpressten Wählers besteht. Er weiß, dass die ehrlichen Wähler gemäß der Verteilung $\vec{p}$ abstimmen. Stimmt der erpresste Wähler ehrlich ab, so sollte das gesamte pure result konsistent mit der Verteilung $\vec{p}$ sein, da der erpresste Wähler in diesem Fall als ehrlicher Wähler agiert und auch gemäß $\vec{p}$ abstimmt. Stimmt der erpresste Wähler nach dem Angreiferwunsch ab, so wird die Verteilung leicht verschoben[1]. Genau das nutzt der Angreifer aus, um zu entscheiden, ob der erpresste Wähler die Ausweichstrategie $\tilde{v_S}$ oder die Dummystrategie *dum* ausgeführt hat.

Sei nun $\vec{r} = (r_0, \ldots, r_k)$ das pure result. Es gilt $r_0 + \ldots + r_k = n + 1$, und r_0 ist die Anzahl Enthaltungen. Da das pure result immer erst nach der Wahl bekannt ist, und es mehrere mögliche pure results gibt, arbeiten wir mit der Menge Res aller möglichen pure results.

Sei $A^i_{\vec{r}}$ die Wahrscheinlichkeit, dass die Entscheidungen der ehrlichen Wähler zum pure result $\vec{r}$ führen, falls der erpresste Wähler Kandidat i wählt. Es gilt

$$A^i_{\vec{r}} = \frac{n!}{r_0! \cdots r_k!} \cdot p_0^{r_0} \cdots p_k^{r_k} \cdot \frac{r_i}{p_i}.$$

Natürlich sollte der Angreifer nach der Wahl einen Lauf nur dann akzeptieren (d.h. den erpressten Wähler bezahlen bzw. nicht bestrafen), wenn gilt $A^i_{\vec{r}} \leq A^j_{\vec{r}}$, d.h. wenn es für das aus dem Wahlergebnis berechnete pure result wahrscheinlicher ist, dass der erpresste Wähler j gewählt hat. Außerdem macht es Sinn für den Angreifer, wenn er keine speziellen Kandidatenwünsche hat, vor der Wahl den Wähler zu dem Kandidaten j anweisen, bei dem für die möglichen pure results in Res die Differenz $A^i_{\vec{r}} - A^j_{\vec{r}}$ durchschnittlich am größten ist. Sei $M_{i,j} = \{\vec{r} \in Res | A^i_{\vec{r}} \leq A^j_{\vec{r}}\}$, also die Menge aller möglichen pure results, für die es wahrscheinlicher ist, dass der erpresste Wähler nicht i sondern j gewählt hat.

Es gilt

$$A^i_{\vec{r}} \leq A^j_{\vec{r}} \Leftrightarrow \frac{r_j}{r_i} \geq \frac{p_j}{p_i},$$

d.h. es ist genau dann wahrscheinlicher, dass der erpresste Wähler j gewählt hat, wenn das Verhältnis der Stimmen für j zu den Stimmen für i im pure result größer ist als der Erwartungswert. Diese Äquivalenz ist leicht nachzuvollziehen, wenn man sich die Formel für $A^i_{\vec{r}}$ anschaut. Damit können wir nun das minimale δ angeben. Es gilt

$$\delta^i_{min}(n, k, \vec{p}) = \max_{j \in \{1,\ldots,k\}} \sum_{\vec{r} \in M_{i,j}} (A^i_{\vec{r}} - A^j_{\vec{r}})$$

ist das minimale δ, das ein Wahlverfahren erreichen kann, wenn der erpresste Wähler Kandidat i wählen möchte. Für einen formalen Beweis für die Minimalität sei auf [KTV12] verwiesen, dort wird bewiesen, dass das ideale Protokoll P_{ideal} δ^i_{min}-nicht-Erpressbar bezüglich γ_i ist.

4.2.2.6 Analyse von Bingo Voting im Küsters-Truderung-Vogt-Modell

$S = P_{Bingo}(k, m, n, \vec{p})$

[1] Aus diesem Grund muss der Angreifer auch die Stimmen der korrumpierten Wähler abziehen und mit dem pure result statt dem Wahlergebnis arbeiten, denn die Stimmen der korrumpierten Wähler entsprechen nicht der Verteilung $\vec{p}$ und würden damit ebenso die Statistik verfälschen.

Teilnehmer

- Voting Authority:
  - Zufallszahlengenerator (TRNG) ehrlich
  - Wahlmaschine ehrlich
  - Bulletin Board ggf. korrumpiert
  - ehrlicher Auditor mindestens einer ehrlich
- ehrlicher Wähler:
  1. Der Wähler entscheidet sich bzgl $\vec{p}$ für einen Kandidaten $v \in [0, k]$
  2. Für $v = 0$ enthält sich der Wähler, sonst gibt er eine Stimme für v ab.
  3. Die Quittung wird bekannt gegeben und geprüft.
- erpresster Wähler:
  - Der erpresste Wähler hat ggf. eine Tonverbindung zum Angreifer aber keine Video oder Bildverbindung. Außerdem verfügt er über einen *untappable channel* zu der Wahlmaschine M und den Zufallszahlengenerator.
  - Wählerwunsch γ_i
  - Gegenstrategie $\tilde{v}_s$

Angreifer Der Angreifer kann unehrliche teilnehmer vollständig kontrollieren und mit dem erpressten Wähler kommunizieren (über eine Tonverbindung). Die Sicht des Angreifers ist wie folgt:

1. sein Zufall (und der vollständig kontrollierten Wähler)
2. den Inhalt des Bulletin Boards, das am Ende der Wahl auch die Wahlergebnisse enthält.
3. alle Quittungen
4. die vom erpressten Wähler an ihn gesendete Nachrichten
5. allei Informationen, die die korrumpierten Parteien besitzen

Gegenstrategie: Wähler tut das gleiche wie in der Dummy-Strategie *dum*, bis auf die folgenden Ausnahmen:

1. die Gegenstrategie $\tilde{v}_s$ wählt für den Kandidaten i statt für j
2. wenn *dum* die Zahl auf dem TRNG weitergeben würde, gibt $\tilde{v}_s$ die Zahl neben j auf der Quittung weiter.

Beweis

1. Der erste Punkt aus der Definition der "nicht-erpressbarkeit" ist erfüllt, da der erpresste Wähler bei der oben beschriebenen Gegenstrategie für den Kandidaten i stimmt, und nie für den vom Angreifer geforderten Kandidaten j.
2. Für den etwa achtseitigen Beweis: siehe Papier [KTV12].

In diesem Modell werden Kollisionen zwischen Zufallszahlen nicht berücksichtigt.

Kapitel 5

Kryptographische Wahlverfahren für Präsenzwahlen

5.1 Bingo Voting

Das Verfahren *Bingo Voting*[BMQR07, Röh07], das am Lehrstuhl IKS Müller-Quade entwickelt wurde, verwendet eine Wahlmaschine und einen vertrauenswürdigen Zufallsgenerator, um ein nachweislich korrektes Wahlergebnis zu erhalten, wobei der Wähler nicht erpresst werden kann. Die Autorisierung der Wähler geschieht z.B. wie in den herkömmlichen, papierbasierten Wahlverfahren.

Bingo Voting wurde im Jahr 2008 erfolgreich eingesetzt bei der Studierendenparlamentswahl der Universität Karlsruhe. Ein kleiner Erfahrungsbericht ist in [BHMQ+08] enthalten.

Das Verfahren ist sehr flexibel und erlaubt z.B. Kumulieren und Panaschieren, also das Verteilen mehrerer Stimmen auf mehrere Kandidaten. Der Einfachheit halber beschreiben wir das Verfahren hier jedoch für eine 1-aus-n-Auswahl, also einer Stimme pro Wähler, die dieser einem von n Kandidaten geben kann.

5.1.1 Prinzip von Bingo Voting

Das Verfahren wird weiter unten noch im Detail beschrieben, dieser Abschnitt gibt einen kurzen Überblick. Vor der Wahl wird für jeden Kandidaten eine Liste mit Zufallszahlen generiert. Die Länge der Liste entspricht mindestens der Anzahl der Wahlberechtigten. Die Zufallszahlen dienen als Füllstimmen, zwischen denen die echte Stimme versteckt werden sollen. In der Wahlphase teilt der Wähler der Wahlmaschine mit, für welchen Kandidaten er stimmen möchte. Nun ordnet die Wahlmaschine diesem Kandidaten eine neue Zufallszahl vom Zufallszahlengenerator zu, während die anderen Kandidaten eine Füllstimme, also eine Zufallszahl aus den vor der Wahl erstellten Liste, erhalten. Keine Zufallszahl darf doppelt verwendet werden. Der Wähler erhält einen Ausdruck der Quittung, die die Kandidaten und die jeweils zugeordneten Zufallszahlen enthält. Er kann die Zufallszahl hinter dem von ihm gewählten Kandidaten mit jener vergleichen, die der Zufallszahlengenerator noch anzeigt. Am Ende der Wahl werden alle Quittungen veröffentlicht. Einer Quittung ist dabei nicht anzusehen, für welchen Kandidaten gewählt wurde. Lediglich mit dem Wissen, welche Zufallszahl der Zufallszahlengenerator angezeigt hat, oder welche Zahlen die Listen enthalten, kann das Wahlgeheimnis gebrochen werden.

Wenn nicht bekannt ist, welche Zufallszahlen vor der Wahl erstellt wurden, sondern nur bewiesen werden kann, dass es sich um genau eine neue und sonst um vor der Wahl erstellte Zufallszahlen handelt, kann bewiesen werden, dass die Auszählung korrekt ist.

Dazu legt sich die die Authority vor der Wahl mit (maskierbaren) Commitments auf die erstellten Zufallszahlen fest. Diese Commitments werden veröffentlicht. Für die neue Zufallszahl wird während der Wahl ebenfalls ein Commitment erstellt. Mit Hilfe eines Shuffles wird gezeigt, dass alle Commitments bis auf eines bereits existierten und jedes zu einer der Zufallszahlen und dem dazugehörigen Kandidaten gehört.

5.1.2 Voraussetzungen

Wahlgeheinnis, Quittungsfreiheit und Nichterpressbarkeit Für das Wahlgeheimnis, und damit auch die Quittungsfreiheit und die Nicht-Erpressbarkeit, muss der Wahlcomputer vertrauenswürdig sein.

Verifizierbarkeit Für die Verifizierbarkeit muss nur der Zufallszahlengenerator vertrauenswürdig sein. Der Wahlcomputer kann manipuliert sein. Der Angreifer sollte jedoch nicht voraussehen können, welcher Wähler seine Stimme überprüft.

Wählerauthentifikation Bingo Voting beschreibt nur die Wahldurchführung selbst, die Wählerauthentifikation und die Anmeldung des Wählers an der Wahlmaschine muss über eine zusätzliche, sichere Methode erfolgen. Etwa kann der Wähler sich wie bei der Papierwahl im Wahlbüro authentifizieren und dann mittels einer anonymisierten Chipkarte, die er vom Wahlvorstand erhält, bei der Wahlmaschine anmelden. Andere Vorgehensweisen sind natürlich denkbar.

5.1.3 Notationen

Es sei l die Anzahl Wähler, und n die Anzahl Kandidaten. Mit *com* bezeichnen wir die *commit*-Funktion eines Commitment-Verfahrens. Für Bingo Voting eignen sich Pederson-Commitments (Abschnitt 2.2.1) sehr gut, da sie perfekt versteckend sind und maskierbar ohne Kenntnis des Commitment-Inhalts.

5.1.4 Teilnehmer

Wahlmaschine/Wahlauthority Die Wahlmaschine muss für das Wahlgeheimnis vertrauenswürdig sein, d.h. die Wahlauthority, die die Wahlmaschine unter Kontrolle[1] hat, muss das auch sein. Wir modellieren dies hier als eine Instanz und nennen sie im Folgenden der Einfachheit halber die Wahlmaschine.

Wähler Geben ihre Stimme an der Wahlmaschine in der Wahlkabine ab, und prüfen wärend und nach der Wahl freiwillig ihre Stimmabgabe und die Auszählung.

Public Bulletin Board Zum Dinge veröffentlichen.

5.1.5 Vor der Wahl

Vor der Wahl werden Füllstimmen erstellt, die später dazu dienen, die echte Stimme auf der Quittung zu verschleiern. Dazu werden für jeden Kandidaten l Zufallszahlen erstellt[2], also pro Kandidat für jeden berechtigten Wähler eine. Die Zufallszahlen müssen alle paarweise verschieden sein. Außerdem dürfen sie nicht von Zahlen des in der Wahl verwendeten Zufallszahlengenerators unterscheidbar sein, idealerweise werden sie aus der gleichen Zufallsquelle gezogen. Die vor der Wahl erstellten Zufallszahlen müssen unbedingt geheim gehalten, jedoch noch vor der Wahl fest den einzelnen Kandidaten zugeordnet werden. Dazu wird für jeden Kandidaten C_i, $i = 1, \ldots, n$ und jede seiner Zufallszahlen $R_{i,j}$, $j = 1, \ldots, l$ ein Commitment auf das Paar $(C_i, R_{i,j})$ erstellt in der Form $c_{i,j} := (com(C_i), com(R_{i,j}))$. Dabei wird für jedes Commitment neuer Commitmentzufall verwendet, sodass je zwei Commitments $com(C_i)$ und $com(C_k)$ nicht anzusehen ist, ob sie den gleichen Kandidaten beinhalten. Wir nennen das Paar $(C_i, R_{i,j})$ eine *Füllstimme* für Kandidat C_i. Die Commitments auf die Füllstimmen werden vor der Wahl auf dem Bulletin Board veröffentlicht, zusammen mit einem Beweis, dass jeder Kandidat gleich viele Füllstimmen bekommen hat, nämlich genau l Stück. Das ist wichtig, weil wir später über die übrigen Füllstimmen die Auszählung berechnen. Der Beweis ist der Grund dafür, warum sich in den Tupeln $(com(C_i), com(R_{i,j}))$ auf Kandidatenname und Zahl einzeln commited wurde, im Beweis verwenden wir nämlich nur jeweils den ersten Teil $com(C_i)$ jedes Tupels. Er funktioniert folgendermaßen: Alle Commitments $com(C_i)$ werden mittels eines verifizierbaren Mix maskiert, gemischt und dann geoffnet. Unter den hierbei geöffneten Kandidatennamen muss nun jeder genau l mal vorkommen. Damit ist bewiesen, dass es zu jedem Kandidatennamen C_i genau l Tupel $c_{i,j} = (com(C_i), com(R_{i,j}))$ gibt, die ihn enthalten, damit gibt es für jeden Kandidaten genau l Füllstimmen. Verschiedene Mix-Techniken haben wir in Abschnit 2.5 kennengelernt. Welche davon hier verwendet wird, spielt hier keine so große Rolle. Eine weitere Beweistechnik, die recht kompliziert, aber dafür effizient ist und jede Manipulation mit 100%iger Wahrscheinlichkeit erkennt, wird in der Dissertation [Hen12] von Christian Henrich vorgestellt. Dort wird auch eine effizientere Variante von Bingo Voting vorgestellt, die den Beweis der gleich vielen Füllstimmen erst nach der Wahlphase macht, wodurch das anfallende Datenvolumen für den Beweis deutlich verringert wird. Dem einfacheren Verständnis zuliebe stellen wir hier jedoch nur die Originalversion vor.

[1] Natürlich kann die Wahlauthority dabei aus mehreren Personen bestehen, von denen keiner allein Zugriff auf die Wahlmaschine hat, die Wahlauthority ist dann als Gruppe vertrauenswürdig.

[2] Kann der Wähler nicht nur eine, sondern k Stimmen vergeben, so werden $l * k$ Zufallszahlen pro Kandidat erstellt.

5.1.6 Wahlphase

In der Wahlkabine findet der Wähler die Wahlmaschine, den Zufallszahlengenerator mit eignem Display und einen Quittungsdrucker vor. Der Wähler teilt der Wahlmaschine seine Auswahl mit. Dabei ist die genaue UI der Wahlmaschine nicht weiter durch das Verfahren eingeschränkt, es kann z.B. ein herkömmlicher Stimmzettel am Bildschirm angezeigt werden, den der Wähler dann per Touchscreen "ankreuzt". Hat der Wähler seine Auswahl getroffen, so wird seine Quittung erstellt. Dazu erzeugt der Zufallszahlengenerator eine frische Zufallszahl. Dem gewählten Kandidaten wird diese frische Zufallszahl zugeordnet, allen anderen, nicht gewählten Kandidaten eine ihrer noch nicht in der Wahlphase verwendeten Füllstimmen (Es ist wichtig, dass Füllstimen wärend der Wahlphase nicht doppelt verwendet werden. Dies kann man später nachprüfen.). Auf die Quittung wird nun für jeden Kandidaten ein Paar (Kandidatenname, Zufallszahl) gedruckt, wobei der gewählte Kandidat mit der frischen Zufallszahl vertreten ist, und die anderen Kandidaten jeweils mit einer Füllstimme. Der Wähler bekommt einen Ausdruck der Quittung, die auch in der Wahlmaschine elektronisch gespeichert wird. Er kann nun prüfen, ob die Wahlmaschine die Auswahl richtig entgegengenommen hat, indem der die Zufallszahl hinter dem gewählten Kandidaten auf der Quittung mit der Zahl auf dem Display des Zufallszahlengenerators vergleicht. Stimmen beide Zahlen überein, so verlässt der Wähler zufrieden die Wahlkabine und das Display des Zufallszahlengenerators wird gelöscht. Nun erkennt niemand außer dem Wähler, welches auf der Quitung die frische Zufallszahl ist, und der Wähler kann sein Wissen darüber auch niemandem beweisen.

5.1.7 Nach der Wahl: Auszählung und Verifikation

Nach der Wahl wird veröffentlicht:

- Das Wahlergebnis
- Alle nicht benutzten Füllstimmen zusammen mit der unveil-Information der entsprechenden Commitments (unmaskiert!)
- Alle Quittungen
- Beweise, dass jede Quittung genau eine frische Zufallszahl und sonst Füllstimmen enthält

Genauer: Das Wahlergebnis wird über die übrigen, nicht in der Wahlphase verwendeten Füllstimmen berechnet: Wärend eines Wahlvorgangs verliert jeder Kandidat außer dem Gewählten eine Füllstimme, die auf der Quittung verwendet wird. Der gewählte Kandidat verliert keine Füllstimme, für ihn wird die frische Zufallszahl verwendet. Da es für jeden Kandidaten für jeden Wähler genau eine Füllstimme gibt, spiegeln die übrigen Füllstimmen genau das Wahlergebnis wieder. Falls die Wahlbeteiligung nicht bei 100% liegt, muss hier natürlich noch ein Offset für die Nicht-Wähler abgezogen werden.

Da die übrigen Füllstimmen nicht auf Quittungen auftauchen und damit niemandes Wahlgeheimnis brechen könnten, werden sie einfach direkt geöffnet. Nun kann man erkennen, von welchem Kandidaten wie viele Füllstimmen übrig sind, wieviele echte Stimmen er also bekommen hat.

Um das Ergebnis vollständig verifizieren zu können, muss noch bewiesen werden, dass in jedem Wahlvorgang genau ein Kandidat gewählt wurde (keine Füllstimme verloren hat) und alle anderen Kandidaten je eine Füllstimme verloren haben. Dazu wird bewiesen, dass jede Quittung genau eine frische Zufallszahl und ansonsten nur Zufallszahlen enthält, für die vor der Wahl ein Commitment erstellt wurde. Dazu wird für den gewählten Kandidaten C_W ein frisches Commitment $com(C_W), com(Z)$ erstellt, wobei Z die frische Zufallszahl aus dem Wahlvorgang[3] ist. Die Commitments auf die auf der Quittung verwendeten Füllstimmen werden nun zusammen mit dem frischen Commitment über einen verifizierbaren Mix geöffnet. So kann jeder prüfen, ob alle bis auf eins der im Beweis verwendeten Commitments vor der Wahl veröffentlicht und nicht als unbenutzt geöffnet wurden, und ob beim öffnen der im Mix maskierten Commitments die richtigen Paare (Kandidat, Zufallszahl) herauskommen, die auch auf der Quittung stehen.

Welche Mixtechnik hier verwendet wird, ist wieder relativ egal, aber falls Randomized Partial Checking (Abschnitt 2.5.2) verwendet wird, muss mehr als einmal gemischt werden, sonst wird im Beweis zuviel Information über die Stimme bekannt (die Hälfte der Kandidaten könnte als nicht gewählt erkannt werden).

[3] Der Wahlmaschine muss diese Zahl dazu noch bekannt sein; Sie kann das Commitment auch direkt beim Wahlvorgang erstellen.

Jeder Wähler kann außerdem nachprüfen, ob seine Quittung veröffentlicht wurde und ein entsprechender Beweis dazu vorliegt.

Zusammen mit der Möglichkeit, in der Wahlkabine das vorhandensein der frischen Zufallszahl an der richtigen Stelle der Quittung zu prüfen, kann also jeder Wähler nachprüfen, dass seine Stimme richtig gezählt wurde (*individuelle Verifizierbarkeit*).

Durch Nachprüfen, dass jedes vor der Wahl veröffentlichte Commitment entweder geöffnet wurde oder in einem der Quittungsbeweise vorkommt, und durch den Beweis, dass auf jeder Quittung nur jeweils die Füllstimme eines Kandidaten[4] nicht verwendet wurde, kann die Korrektheit des Wahlergebnisses überprüft werden (*universelle Verifizierbarkeit*).

5.1.8 Vor- und Nachteile des Verfahrens

5.1.8.1 Sicherheit

Individuelle und universelle Verifizierbarkeit sind wie oben beschrieben gegeben. Um Ballot Stuffing zu vermeiden, muss bekannt sein, wie viele Wähler tatsächlich eine Stimme abgegeben haben. Da es sich um ein Präsenzwahlverfahren handelt, kann dies durch Wahlbeobachtung überprüft werden.

Weiß ein Angreifer, welche Wähler ihre Stimme nach der Wahl nicht überprüfen werden, z.B. weil die entsprechenden Quittungen im Mülleimer des Wahlbüros liegen, so könnte er gezielt an dieser Stelle nachträglich die Beweise manipulieren. Doch dieses Problem hat eine Lösung, die in [BHK+09] vorgestellt wird: Werden alle Quittungen über eine sogenannte *Hashkette* verbunden, d.h. wird auf jede Quittung der Hashwert der letzten Quittung gedruckt und dieser Hashwert beim Berechnen des Hashes der aktuellen Quittung mit eingerechnet, so wäre für die nachträgliche Manipulation eine Manipulation aller folgenden Quittungen notwendig, weil sonst der Hashwert auf allen folgenden Quittungen nicht mehr stimmen würde. Wird nur eine der folgenden Quittungen überprüft, fällt die Manipulation auf (die eigene Quittung stimmt nicht mit der veröffentlichten überein). Wird zusätzlich eine digitale Signatur der Wahlmaschine auf jede Quittung geschrieben, so kann der Wähler eine Manipulation auch beweisen.

5.1.8.2 Benutzbarkeit

Der Wahlvorgang selbst unterscheidet sich für den Wähler nicht sehr von einem Herkömmlichen (sofern er einen Wahlcomputer gewohnt ist; Der Bildschirm kann wie gesagt einfach einen herkömmlichen Wahlzettel anzeigen und ein bisschen Hilfestellung dazu geben). Der einzige Unterschied ist, dass der Wähler nun freiwillig zwei Zufallszahlen vergleichen kann, um seine Stimme zu prüfen.

Mit der Nachvollziehbarkeit der Wahl ist es allerdings etwas komplizierter. Das Verfahren ist wie einige kryptographische Wahlverfahren leider für Laien vermutlich schwer verständlich. Allerdings können alle geführten Beweise auch von nicht an der Wahl beteiligten Instanzen nachvollzogen werden, so können unabhängige Institutionen die Beweise prüfen. Die einzelnen Wähler müssen die komplizierten Beweise also nicht unbedingt selbst prüfen, wenn sie darauf vertrauen, dass bei den prüfenden Instanzen eine dabei ist, die kompetent und ehrlich ist.

[4] Ob es die richtige war, konnte der Wähler in der Wahlkabine sehen.

Kapitel 6

Kryptographische Wahlverfahren für Internetwahlen

Internetwahlen bieten gegenüber der Präsenzwahl einige Vorteile: Die Stimme kann von überall aus abgegeben werden, der Wähler muss sich zur Stimmabgabe also nicht an einem bestimmten Ort einfinden. Der Wähler kann bequem von zuhause aus wählen, oder auch von einem beliebigen Urlaubsort (Sofern der Wahlserver nicht IP-Adressen sperrt, die auf Regionen außerhalb des Wahlbezirks hinweisen). Dadurch besteht Hoffnung auf eine höhere Wahlbeteiligung.

Internetwahlen werden in Europa bereits teilweise für Parlamentswahlen eingesetzt, z.B. in Norwegen (siehe Abschnitt 6.3) oder Estland. Aber nicht nur für Parlamentswahlen sind Internetwahlen nützlich: Ihr größtes Einsatzgebiet sind weit weniger sicherheitskritische Wahlen wie nicht-politische Online-Abstimmungen, bei denen vermutlich in vielen Fällen weniger Motivation zur Manipulation besteht. Einige Internetwahlverfahren wie z.B. Helios (siehe Abschnitt 6.1) widmen sich speziell diesem Einsatzzweck und treffen entsprechend weniger harte Maßnahmen gegen die Erpressbarkeit.

Die Tatsache, dass die Stimmabgabe über das Internet erfolgt, stellt uns vor viele zusätzliche Herausforderungen, einige davon sind hier aufgelistet:

- Keine Wahlkabine: Der Wähler wählt in unsicherer Umgebung, damit Schutz vor Erpressung schwerer, Angriffe wie “Family Voting” (Familie steht beim Wählen dahinter) werden möglich.
- Wahl am eigenen Rechner:
  - PC leicht mit Viren/Trojaner infizierbar
  - Stimme kann evtl. abgehört oder manipuliert werden
  - Stimmzetteldarstellung kann manipuliert werden
  - Wähler besitzt evtl. nicht die Expertise, seinen Rechner sicher aufzusetzen und zu schützen, oder die Wahlsoftware korrekt zu installieren
- Internet als unsicherer Übertragungskanal (statt persönlicher Stimmabgabe per Hand in die Urne):
  - Wähler/Wahlsoftware ist für die korrekte Stimmverschlüsselung verantwortlich.
  - Authentifikation: Wähler nicht persönlich mit Ausweis im Wahllokal
  - Schlüssel/Passwort-Austausch für die Wählerauthentifikation
  - Authentifikation des Wahlservers beim Wähler
  - Wähler muss sich überzeugen können, dass seine Stimme korrekt beim richtigen Wahlserver angekommen ist
- Wahlserver online: Angriffe sind Grossflächig und von überall aus möglich, nicht nur vor Ort an einer Wahlmaschine

Dabei werden von kryptographischen Wahlverfahren einige der Probleme nicht direkt gelöst sondern geschickt umgangen: Mit Code Voting lernt der PC z.B. die Stimme nicht und damit ist ein Abhören der Wählereingabe sinnlos. Kann der Wähler durch Einsatz eines Bulletin Boards seine Stimmabgabe prüfen, so wird die Authentifikation des Wahlservers beim Wähler hinfällig: Der Wähler sieht, ob seine Stimme bei den auszuzählenden Stimmen auftaucht. Weitere Beispiele finden sich in den vorgestellten Verfahren selbst, man muss nur genau hinschauen.

6.1 Helios

Helios [Adi] ist ein verifizierbares Internetwahlverfahren, das es sich zum Ziel gesetzt hat, verifizierbare Wahlverfahren einer breiten Zielgruppe zugänglich und verständlich zu machen. Das Verfahren ist deshalb recht einfach gehalten, ist dafür aber in seiner Originalversion nur für Wahlen geeignet, für die keine zu hohen Sicherheitsanforderungen gelten. Helios legt seinen Schwerpunkt auf die Verifizierbarkeit, die als Nebeneffekt einen Integritätsschutz liefert, der ja auch erwünscht ist, wenn niemand die Wahl manipulieren will. Diese ist bei Helios allerdings nur dann gewährleistet, wenn ausreichend viele Wähler und auch nicht wählende Wahlberechtigte ihre Stimmabgabe (bzw. nicht-Stimmabgabe) prüfen. Die Autoren von [Adi] sagen selbst, dass Helios für Wahlen mit “low coercion risk" gedacht ist, also Wahlen, bei denen es keine große Motivation gibt, Wähler zu erpressen. Um dies zu verdeutlichen, gab es in der Originalversion noch einen “Coerce me!”-Button, mit dem man seine Zugangsdaten und weitere Information per Mail an eine beliebige Person schicken konnte, dieser ist aber in neueren Versionen nicht mehr vorhanden. Ein Beispiel für solche low-coercion-risk-Wahlen wären Wahlen in Online-Communities, in denen sich die Wähler sowieso nicht persönlich kennen. Eine Implementierung von Helios steht seit 2008 auf `http://heliosvoting.org` zu Verfügung, hier kann jeder - sofern er einen Google/Yahoo/Facebook/Twitter-Account oder ähnliches hat - eine Wahl anlegen oder in einer angelegten Wahl wählen (sofern er für diese Wahl wahlberechtigt ist). Helios wurde z.B. 2009 in einer speziell angepassten Version bei der Wahl des Universitätspräsidenten in Louvain eingesetzt.

Es gibt mehrere Versionen und Weiterentwicklungen von Helios. In der Originalversion läuft die Wahl auf einem Helios-Server. Der Wähler gibt seine Stimme per Internetbrowser ab, der Server verschlüsselt die Stimme und besitzt auch den geheimen Schlüssel zum Entschlüsseln. Daher muss dem Helios-Server für das Wahlggeheimnis vertraut werden. Ausgezählt wird vom Helios-Server per verifizierbarem Mix, für die Integrität muss man dem Server nicht vertrauen. Das Verfahren erfüllt sowohl individuelle als auch universelle Verifizierbarkeit, wobei je nach Ausführung Ballot Stuffing möglich sein kann, dazu später mehr.

In weiteren Versionen des Verfahrens kann die Auszählung auf mehrere Server verteilt werden. Weiterhin wird in späteren Versionen homomorphe Stimmauszählung verwendet. Auch wird die Wahl mit Pseudonymen statt Wählernamen unterstützt, um das Wahlgehemnis weiter zu schützen. Je nach Version erfolgt die Verschlüsselung der Stimme am eigenen PC oder wird (für den Wähler verifizierbar) vom Helios-Server durchgeführt.

Wir beschreiben hier die in [Adi] vorgestellt Originalversion.

6.1.1 Voraussetzungen

Wahlgeheimnis und Quittungsfreiheit Für das Wahlgeheimnis muss man dem Helios-Server vertrauen.

nicht-Erpressbarkeit wird nicht erfüllt.

Verifizierbarkeit Für die Verifizierbarkeit braucht es keine vertrauenswürdige Instanz, allerdings müssen ausreichend viele Wähler ihre Stimme prüfen. Ballot Stuffing wird nur dann verhindert, wenn nicht-Wähler ihre Stimmabgabe prüfen. In der Version mit Pseudomnymen wird Ballot Stuffing höchstens dann verhindert, wenn ein vertrauenswürdiger Wahladministrator alle Pseudonyme kennt und prüft, ansonsten kann man nicht nachprüfen, ob Pseudonyme zu wahlberechtigten Personen gehören.

6.1.2 Teilnehmer

Bei dieser Wahl gibt es nicht besonders viele unterschiedliche Teilnehmer.

Helios-Server Der Helios-Server nimmt die Stimme entgegen, verschlüsselt sie für die Verifizierbarkeitsbeweise und führt die Auszählung durch.

Wähler Der Wähler gibt seine Stimme ab und überprüft die Integrität der Wahl.

Weiterhin gibt es ein *Bulletin Board* mit den Eigenschaften wie gehabt.

6.1.3 Vor der Wahl

Vor der Wahl sind keine Vorberechnungen nötig. Die Wahl wird aufgesetzt und die Zugangsdaten werden an die wahlberechtigten Personen verteilt. In der Originalversion passiert dies noch per E-Mail, in späteren Versionen werden hier bereits vorhandene Single Sign-On-Accounts der Wähler verwendet. Momentan unterstützt werden dabei Twitter, Google, Facebook und Yahoo.

6.1.4 Wahlphase

In der Wahlphase gibt der Wähler per Internetbrowser als Thin Client seine Stimme ab, die Wahlsoftware läuft auf dem Helios-Server. Es bezeichne E die El-Gamal-Verschlüsselung und H eine öffentlich bekannte kryptographische Hashfunktion. Es werden folgende Schritte durchlaufen:

1. Der Wähler füllt per Internetbrowser seinen Stimmzettel aus, diesen bezeichnen wir ab hier mit v.

2. Der Helios-Server verschlüsselt die Stimme v zu $c := E(v)$ und committet sich auf das Chiffrat, indem er den Hashwert $h := H(c)$ am Wähler-Bildschirm anzeigt.

3. Der Wähler hat nun die Wahl, seinen Stimmzettel zu prüfen.
   kein Test: Entscheidet sich der Wähler, nicht zu prüfen, so geht es weiter bei Schritt 4.
   Test: Im Test-Fall zeigt der Server das Chiffrat c sowie den Zufall für die Verschlüsselung an. Damit kann der Wähler nachprüfen, dass $c = E(v)$ und $h = H(c)$, und damit, dass v korrekt verschlüsselt wurde. Der Wähler wäre nun erpressbar, wenn er c abgeben dürfte, da er durch die offengelegten Werte beweisen könnte, was er gewählt hat. Und - auch, wenn Helios für Wahlen mit niedrigen Sicherheitsanforderungen gedacht ist - will das Verfahren die Erpressbarkeit ja nicht provozieren. Deshalb geht es nun wieder bei Schritt 1 los, evtl. mit gleichem v aber anderem Verschlüsselungszufall.

4. Der Wähler *versiegelt* den Stimmzettel, indem er die Stimmabgabe bestätigt. Der Helios-Server löscht alle in dieser Stimmabgabe verwendeten Zufälle und Klartext-Informationen, nur das (aktuellste, nun ungetestete) Chiffrat c wird auf dem Server gespeichert.

5. Erst jetzt authentifiziert sich der Wähler, und seine Wahlberechtigung wird vom Helios-Server überprüft. Bei Erfolg wird c als abgegebene Stimme auf dem Server gespeichert, das Tupel (ID, c) wird auf dem Bulletin Board veröffentlicht, wobei ID den Wählernamen (oder sein Pseudonym) darstellt.

Dadurch, dass das Authentifizieren erst nach der Testphase stattfindet, kann jeder die Verschlüsselung prüfen, auch nicht-wahlberechtigte Personen. Außerdem kann der Wahlserver so nicht gezielt die Stimmen gezielter Wähler manipulieren (vorausgesetzt, er erkennt sie nicht an ihrer IP-Adresse).

6.1.5 Nach der Wahl

Nach der Wahl passiert folgendes:

1. Der Helios-Server mischt die Stimmen mit einem verifizierbaren Mix, und beweist das korrekte Mischen (auf dem Bulletin Board). Er nimmt dabei den Reencryption-Mix, den wir in Abschnitt 2.5.1 bereits kennengelernt haben, mit der ElGamal-Wiederverschlüsselung (siehe Abschnitt 2.4.4.1), allerdings entschlüsselt er die gemischten Stimmen noch nicht.

2. Die Wähler haben nun für einen bestimmten Zeitraum die Möglichkeit, das korrekte Mischen zu prüfen und sich zu beschweren. Kommt keine gerechtfertigte Beschwerde, so werden die Stimmen geöffnet und ausgezählt, außerdem wird die korrekte Entschlüsselung mit dem Chaum-Pedersen-Protokoll bewiesen, siehe Abschnitt 2.4.4.2. Auszählung und Beweise werden auf dem Bulletin Board veröffentlicht.

3. Jeder kann die Auszählung und alle veröffentlichten Beweise prüfen. Nicht-Wähler sollten spätestens hier prüfen, dass keine Stimme unter ihrem Namen abgegeben wurde.

Es wird an dieser Stelle zum Verständnis empfohlen, sich die in Schritt 1 und 2 referenzierten Abschnitte nochmals anzusehen.

6.1.6 Vor- und Nachteile des Verfahrens

Sicherheit Ein Vorteil des Verfahrens ist, dass jeder die Stimmabgabe überprüfen kann, auch nicht-Wahlberechtigte. Dies wird dadurch ermöglicht, dass Wähler sich erst kurz vor der Stimmabgabe, nach der Testphase authentifizieren müssen.

Das Verfahren sollte allerdings wirklich nur für sicherheits-unkritische Wahlen verwendet werden: Sowohl der Helios-Server als auch der Rechner, an dem die Stimme abgegeben wird, kennen die Stimme im Klartext. Die beiden Geräte als vertrauenswürdig anzunehmen, sollte man sich im Allgemeinen nicht trauen: Ein PC kann leicht von Viren und Trojandern befallen sein, auch von solchen, die die Wählerstimme abhören. Der Wahlserver hat evtl. kompetentere Administratoren und kann besser geschützt werden, jedoch ist er ans Internet angeschlossen und kann entsprechend großflächig angegriffen werden. Das Wahlgeheimnis ist damit bei Helios relativ leicht zu brechen, die Integrität bleibt allerdings bis auf Ballot Stuffing auch bei korrumpiertem Server gewahrt. Bei einem möglicherweise korrumpierten PC müsste in der Testphase noch geprüft werden, dass zum Verifizieren der korrekten Verschlüsselung wirklich das richtige Verifikationsprogramm läuft, oder die Verschlüsselung an einem weiteren Gerät geprüft werden. Falls nicht-Wähler ihre Stimmabgabe nicht prüfen oder Pseudonyme verwendet werden, sind leicht Ballot-Stuffing-Angriffe von der Serverseite aus möglich, vor Allem wenn die Wähler sich nicht untereinander kennen.

Benutzerfreundlichkeit Da alle Überprüfungen optional sind, ist das Verfahren recht benutzerfreundlich. Sollte allerdings der kryptographisch nicht gewandte Wähler auf Test klicken, könnte ihn das Verfahren sehr verwirren. Auch die Tatsache, dass der Wähler nicht *seine* Stimme, sondern nur die Teststimmen überprüfen kann, könnte verwirrend und verunsichernd sein. Außerdem wird die Verifizierung stark empfohlen, die Integrität des Verfahrens hängt sogar davon ab, dass genug Wähler ihre Stimme überprüfen. Es werden von Helios Programme zur Verifikation bereitgestellt. Will man diesen nicht vertrauen, so kann man sie selbst nachimplementieren. Mit um dies zu erleichtern, wurde das Verfahren so einfach gehalten.

6.2 Code Voting

6.2.1 Code Voting

Wie bereits erwähnt haben Onlinewahlverfahren im Vergleich zu in Wahlbüros eingesetzten Wahlmaschinen den Nachteil, dass eine *Wahlmaschine*, die sich bei den Wählern zu Hause befindet[1], schwieriger gegen Korruption zu schützen ist, als ein evtl. spezialisierte Hardware, die sich in einer kontrollierten Umgebung befindet und deren einziger Zweck die Entgegennahme von Wählerstimmen ist. Es muss bei Onlinewahlverfahren also davon ausgegangen werden, dass der Wähler-PC korrumpiert ist. Damit ist selbstverständlich, dass bei einem idealen Onlinewahlverfahren keines der Geräte, die der Wähler zur Stimmabgabe einsetzt, die Stimme des Wählers in Erfahrung bringen darf. Ebenso wenig darf es ihm Möglich sein, die Stimme unbemerkt zu verändern.

Code Voting ist eine Methode, deren Ziel es ist, zu verhindern, dass aus der Eingabe des Wählers ohne zusätzliches Wissen der Kandidat abgeleitet werden kann, für den der Wähler tatsächlich stimmt. Dazu erhält der Wähler eine Abbildung von den Kandidaten auf *Codes*. Statt des Kandidatennamens gibt der Wähler den zu dem Kandidaten gehörenden Code ein. Der PC selbst lernt dem zufolge nur den Code. Schwieriger ist, wie aus dem Code wieder der Kandidatenname abgeleitet werden kann, ohne dass ein anderes System die Wahl dieses einen Wählers kennen lernt, da für jeden Wähler eine andere Abbildung gewählt werden muss. Erhielte jeder Wähler die gleiche Abbildung, würde sie mit überwältigender Wahrscheinlichkeit auch der Angreifer und somit der korrumpierte PC des Wählers kennen.

Oft werden in Kombination mit Code Voting auch Return Codes eingesetzt. Bei diesem Verfahren wird dem Wähler – in der Regel auf einem zweiten Kanal[2] – ein Zufallscode als Antwort auf seine abgegebene Stimme geschickt. Anhand dieser Antwort kann der Wähler erkennen, ob die Stimme von seinem PC wie vorgesehen weitergeleitet wurde. Das bedeutet, dass für jeden Kandidat ein eigener Return Code vergeben wurde. Ähnlich wie bei den Codes des Code Voting müssen sich auch die Return Codes von Wähler zu Wähler unterscheiden. Damit der Wähler die Return Codes kontrollieren kann, werden sie ihm zusammen mit den Codes für das Code Voting auf der Wahlbenachrichtigung mitgeteilt.

[1] PC, Handy, ...
[2] z. B. per SMS

6.3 E-Valg Norwegen

Im Jahr 2009 hat sich die Regierung Norwegens dafür entschieden, für die Gemeindewahlen auch ein Onlinewahlverfahren einzusetzen. Das Verfahren sieht eine mehrwöchige Onlinewahlphase vor, in der jeder Wähler mehrfach eine Stimme abgeben kann. Die jeweils letzte Stimme jeden Wählers geht in das Wahlergebnis ein. Zusätzlich zu der Onlinewahl gibt es nach deren Ende noch eine klassische Papierwahlphase. Stimmen, die im klassischen Verfahren abgegeben werden, überschreiben die Onlinestimmen. Für die Abgabe und Auszählung der Onlinestimmen wird das eigends für die norwegische Wahl entwickelte Verfahren E-Valg [Gjø10] verwendet.

6.3.1 Revoting

Onlinewahlsysteme haben den Vorteil, dass der Wähler seine Stimme an einem Ort seiner Wahl abgeben kann. Durch die damit verbundene Zeitersparnis für den Wähler erhofft man sich eine höhere Wahlbeteiligung. Ein grundsätzliches Problem, das mit diesem Vorteil einhergeht, ist der Umstand, dass die Stimmabgabe nicht mehr in einer kontrollierten Umgebung statt findet. Der Gesetzgeber hat, wie es in Kapitel 1.2.3 beschrieben wird, Vorgaben für den Ablauf einer Wahl festgelegt, die sicherstellen sollen, dass der Wähler bei seiner Stimmabgabe nicht beeinflusst wird. Dies wird im wesentlichen durch die Wahlzelle erreicht, in der der Wähler, durch den Wahlvorstand kontrolliert, alleine das Kreuz auf seinem Stimmzettel setzt.

Die Garantie, dass der Wähler unbeobachtet seinen Stimmzettel ausfüllt, kann bei Onlineverfahren genauso wenig gegeben werden, wie bei der Briefwahl. Im Gegensatz zur Briefwahl, soll ein Onlinewahlsystem nicht eine Alternative für Ausnahmefälle sein. Dadurch würde einem Angreifer die Möglichkeit eröffnet, regulär wählende Bürger zu erpressen, indem er während der Stimmabgabe anwesend ist und diese in seinem Interesse steuert. Da das prinzipiell nicht zu verhindern ist, wird eine Methode benötigt, die diesen Angriff – wenn auch nicht verhindern – zumindest abschwächen kann.

Aus diesen Gründen soll es dem Wähler möglich sein mehrfach eine Stimme abzugeben. Dabei soll jeweils die letzte Stimme in das Wahlergebnis eingehen, während die zuvor abgegebenen keinen Einfluss mehr haben. Das ermöglicht einem erpressten Wähler, dass er in einem unbeobachteten Moment nach der Erpressung seine eigene Stimme abgeben kann, und die durch die Erpressung erzwungene Stimme ihre Gültigkeit verliert. Je länger die Wahlphase ist, desto höher ist die Wahrscheinlichkeit, dass es einen solchen Moment gibt. Natürlich ist es denkbar, dass der Angreifer den Wähler erst nach dem Ende der Wahlphase wieder unbeobachtet lässt. Im Allgemeinen geht man aber davon aus, dass ein solcher Angriff nicht im großen Rahmen durchführbar ist, da pro Wähler ein Angreifer benötigt wird.

Welche Bedingungen müssen gelten, damit Revoting tatsächlich die erhofften Vorteile bringt?

- Zunächst darf der Angreifer nicht beeinflussen können, ob der Wähler erneut eine Stimme abgibt. Das bedeutet insbesondere, dass die erneute Stimmabgabe nicht von Details der ersten Stimmabgabe abhängen darf. Ist beispielsweise die Kenntnis einer bei der erpressten Stimmabgabe verwendeten Zufallszahl notwendig, ist es dem Angreifer möglich dieses Wissen *mitzunehmen* und so dem Wähler die Möglichkeit der erneuten Stimmabgabe zu berauben.
- Die Zahl der erneuten Stimmabgaben darf auch nicht nach oben beschränkt sein. Gäbe es eine obere Schranke, könnte der Angreifer mit dem Wähler so oft Stimmen abgeben, bis diese Grenze erreicht ist.
- Aus der Zahl der überschriebenen Stimmen kann ein Angreifer ebenfalls herauslesen, ob seine Stimme überschrieben wurde. Eine Zuordnung, welche (verschlüsselte) Stimme von welchem Wähler stammt, darf vor dem Aussortieren der überschrieben Stimmen nicht sichtbar werden.
- Zuletzt darf die Stimme kein Merkmal enthalten, die es dem Angreifer ermöglicht zu erkennen, ob die in seiner Anwesenheit abgegebene Stimme in die Auszählung mit eingeht.

Die Anforderungen, die bisher an Onlinewahlsysteme gestellt wurden, müssen, wenn Revoting zum Einsatz kommt, durch weitere Details ergänzt werden. Kurz zusammengefasst: Es muss dem Wähler gezeigt werden, dass die Stimme, die *er* zuletzt abgegeben hat, in das Wahlergebnis eingeht. Dem Wähler soll also gezeigt werden, dass ...

... alle überschrieben Stimmen nicht gezählt werden.
... seine letzte Stimme von niemandem überschrieben wurde.
... eine seiner Stimmen in das Ergebnis mit eingeht.

Dabei darf dem Angreifer – wie bereits erwähnt – nicht gezeigt werden, dass seine erpresste Stimme nicht Teil des Wahlergebnisses ist.

Die bisher formulierten Forderungen beschreiben lediglich die individuelle Verifizierbarkeit. Wünschenswert wäre hingegen die universelle Verifizierbarkeit, die es jedem (Wähler und Nichtwähler) ermöglicht die Beweise nachvollziehen zu können. Für die universelle Verifizierbarkeit sollte außerdem noch gezeigt werden, dass jede Stimme, die in das Wahlergebnis eingeht, von einem legitimen Wähler stammt. Dieses Problem hängt zwar eng mit dem Revoting-Prozess zusammen, ist aber eigentlich Teil des *Urnenbuchproblems*.

Bisher[3] erfüllt keines der bekannten und in der Vorlesung vorgestellten Protokolle die oben beschriebenen Anforderungen. Bisher wird in den Verfahren entweder ein bestimmter Teilnehmer als vertrauenswürdig angenommen, der dafür sorgt, dass kritische Informationen nicht an den Angreifer gelangen können, oder es werden nicht alle Ziele erreicht.

6.3.2 Prinzip von E-Valg

Das Protokoll und die Software ist zwar für alle einsehbar, jedoch bietet dieses Protokoll für sich genommen keinen Schutz gegen alle Angriffsszenarien, die ausgeschlossen werden sollen. Dazu ist es erforderlich, dass verschiedene Komponenten nicht korrumpiert sind und nicht zusammenarbeiten. Daher werden die Server auf mehrere, räumlich getrennte Rechenzentren aufgeteilt.

6.3.3 Teilnehmer

Wäher-PC: verschlüsselt und signiert die Wählerstimme
Ballot Box: nimmt verschlüsselte Stimmzettel entgegen und speichert sie
Receipt-Code Generator: erzeugt den Receipt-Code
Decryption Service: entschlüsselt die Stimmzettel und zählt sie aus
Electronial Board: erzeugt die benötigten Schlüssel
Auditoren: zur Kontrolle der Wahl

6.3.4 Kommunikationswege

Post: Der Wähler erhält die Codekarte mit den Kandidatennamen und den Receipt-Codes per Post.

Internet: Der PC des Wählers sendet die verschlüsselte Stimme über einen verschlüsselten Kanal zur Ballot Box.

SMS: Der Wähler erhält den berechneten Receipt-Code, der zu der Stimme gehört, die der PC für ihn abgegeben hat, per SMS.

6.3.5 Voraussetzungen

Authentifizierung: Bei den norwegischen Wahlen konnten sich die Wähler mit Hilfe der folgenden Verfahren ausweisen:
- MinID
- Buypass
- Commfides

Wahlgeheimnis und Verifizierbarkeit: Vollständige öffentliche Verifizierbarkeit ist bei EValg nicht gegeben, die Wähler können lediglich ihre Receipt-Codes prüfen. Alles andere wird nur von Auditoren geprüft. Ansonsten gelten folgende Voraussetzungen:
- Der Wähler-PC ist unkorrumpiert (er lernt und verschlüsselt die Wählerstimme).
- Von den Wahlservern ist nicht mehr als eine Komponente korrumpiert.
- Die Auditoren sind als Gruppe vertrauenswürdig.

6.3.6 Ablauf der Wahl

6.3.6.1 Pre-Election

Vor der Wahl müssen die Schlüsselpaare für die einzelnen Komponenten und die Receipt-Codes für jeden Wähler generiert werden. Sei dazu G eine zyklische Gruppe mit Primzahlordnung p und g ein Erzeuger dieser Gruppe.

1. Das Electronical Board generiert die ElGamal-Schlüsselpaare:

[3] Stand Juni 2013

- $(pk_{DS},\ sk_{DS})$ für den Decryption Service $(pk_{DS} = g^{sk_{DS}})$,
- $(pk_{BB},\ sk_{BB})$ für die Ballot Box $(pk_{BB} = g^{sk_{BB}})$,
- $(pk_R,\ sk_R)$ für den Receipt-Code Generator $(pk_R = g^{sk_R})$.

Die Schlüsselpaare werden so generiert, dass $sk_{DS} + sk_{BB} \equiv sk_R \mod p$, damit gilt $pk_{DS} * pk_{BB} \equiv pk_R$.

2. Für jeden Wähler wird eine Pseudozufallsfunktion d_i erstellt (für Receipt-Codes).

3. Ein Geheimnis $s_i \in \{0, \ldots, p-1\}$ wird für jeden Wähler W_i gewählt. Dieses Geheimnis s_i ist den Wählern selbst nicht bekannt.

4. Für jeden Wähler W_i wird die Receipt-Code-Function r_i, die die Menge der Kandidaten auf die Menge der Receipt-Codes abbildet, bestimmt: Für jede Wählerstimme v gilt $r_i(v) := d_i(v)^{s_i}$.

5. Jeder Wähler W_i erhält eine Code-Karte, auf der der individuelle Receipt-Code des Wählers (bestimmt durch das entsprechende r_i) zu jedem Kandidaten enthalten ist.

6. Die Ballot Box bekommt alle s_i.

7. Der Receipt-Code-Generator bekommt alle d_i.

6.3.6.2 Wahlvorgang

1. Anmelden und Authentifizieren:
   - Der Wähler authentifiziert sich bei der Ballot Box (z. B. mit Hilfe der MinID)
   - Die Ballot Box prüft die Wahlberechtigung.
   - Der Wähler erhält den Public Key des Decryption Services pk_{DS}

2. Der Wähler gibt seine Stimme ab:
   - Der Wähler gibt an seinem PC seine Stimme v_i ein.
   - PC verschlüsselt die Stimme:

     $$E(v_i) := (\alpha_i,\ \beta_i) = (g^{t_i},\ pk_{DS}^{t_i} \cdot v_i)$$

     Dabei ist t_i der verwendete Zufall.
   - Der PC signiert $E(v_i)$ und erstellt einen Beweis $\mathcal{P}$ über die Kenntnis des enthaltenen Klartextes[4], für den Beweis wird das in Abschnitt 2.4.4.3 beschriebene Schnorr-Protokoll verwendet.

3. Verarbeitung durch die Ballot Box
   - Die Ballot Box überprüft den Beweis $\mathcal{P}$.
   - Ist der Beweis korrekt, speichert sie

     $$(W_i,\ SN_i,\ E(v_i))\,,$$

     wobei SN_i eine Seriennummer der abgegeben Stimme ist, um später die Reihenfolge, in der die Stimmen abgegeben wurden, rekonstruieren zu können. Statt einer Seriennummer ist auch ein eindeutiger Zeitstempel denkbar.

   - Die Ballot Box berechnet

     $$(\bar{\alpha}_i,\ \bar{\beta}_i) = (\alpha_i^{s_i},\ \beta_i^{s_i} \cdot \alpha_i^{s_i \cdot sk_{BB}})$$

   - Die Ballot Box sendet

     $$\big(W_i,\ SN_i,\ (\bar{\alpha}_i,\ \bar{\beta}_i)\big)$$

     mit einem Beweis für die korrekte Berechnung an den Receipt-Code-Generator.

4. Verarbeitung durch den Receipt-Code-Generator

[4] Ohne diesen Beweis könnten neue Stimmen mit alten überschrieben werden, ohne den geheimen Schlüssel des Wählers zu kennen (Replay-Attacke).

- Der Receipt-Code-Generator speichert die Informationen.
- Der Receipt-Code-Generator berechnet den Receipt-Code:

$$r_i = d_i\left(\frac{\bar{\beta}_i}{\bar{\alpha}_i{}^{SK_R}}\right) = d_i(v_i^{s_i})$$

(Nachrechnen, dass der Receipt-Code dem entspricht, was der Wähler auf seiner Code-Karte erhalten hat, wird als Übung empfohlen. Zur Erinnerung: $sk_{DS} + sk_{BB} \equiv sk_R$)

5. Der Receipt-Code-Generator sendet r_i an den Wähler per SMS
6. Der Wähler prüft r_i mit Hilfe seine Code-Karte.
7. Optional kann der Wähler seine *Onlinestimme* noch mit einer klassischen *Papierstimme* überschreiben.

6.3.6.3 Auszählung

1. Ballot Box verwirft anhand des Eintrags W_i und der Sequenznummer alle überschriebenen Stimmen.
2. Alle Daten werden von der Ballot Box und dem Receipt-Code-Generator auf externe Datenträger übertragen und von den Auditoren überprüft.
3. Der Decryption-Service bekommt die Chiffrate und führt ein Re-Encryption-Mix mit Beweis durch. Der Mix ist nicht öffentlich verifizierbar. Der Beweis ist nur für die Auditoren einsehbar.
4. Der Decryption-Service entschlüsselt die Stimmen und beweist das korrekte Entschlüsseln. Auch die Entschlüsselung findet verdeckt und nur für die Auditoren nachvollziehbar korrekt statt.
5. Die Auditoren prüfen die Beweise des Mixes und der Entschlüsselung.
6. Das Wahlergebnis wird veröffentlicht.

6.3.6.4 Schwächen des Systems

- Das Wahlgeheimnis besteht nur serverseitig, nicht clientseitig: Der Wähler-PC lernt die Stimme.
- Wähler dürfen nicht mit dem Handy wählen.
- Der Wähler kann mit seinem Handy (SMS) und der Codekarte beweisen, was er gewählt hat. Der Angriff wird durch Revoting zwar theoretisch abgeschwächt, wenn der Wähler die richtige SMS löscht, kann aber in der Praxis eine Gefahr sein.
- Betrug wird vom Wähler durch den falschen Receipt-Code bemerkt, ist aber nicht beweisbar. Der Wähler kann nur erneut wählen.

6.4 Polyas

Das Onlinewahlverfahren Polyas ist nur für Administratoren der Server verifizierbar. Es wurde für Wahlen entworfen, bei denen es nur geringe Sicherheitsrisiken gibt. Wahlen in Vereinen oder ähnlichen Strukturen sind ein Beispiel. Beim Entwurf wurde auf die Praxistauglichkeit geachtet. Insbesondere auf die Effizienz wurde Wert gelegt. Das Papier beschreibt daher auch viele Umsetzungsdetails, was in akademisch motivierten Entwürfen eher unüblich ist. Polyas erfüllt die common criteria Schutzprofil für Online-Wahlprodukte[BSI08]. Es wird vom DFG und der GI eingesetzt. Die Sicherheit beruht auf den Prinzipien *seperation of duties* und dem *viele-Augen-Prinzip*.

6.4.1 Teilnehmer

6.4.1.1 Personen:

- Wähler, der die Stimme an seinem PC eingibt
- Wahlvorstand (Election Commitee): Serveradministratoren, Auditoren, ...
  Sie bilden durch gegenseitige Kontrolle eine vertrauenswürdige Gruppe.

6.4.1.2 Clientseitige Teilnehmer:

- PC des Wählers
- Vote Casting Interface (VCI), zur Stimmabgabe

6.4.1.3 Wahlserver:

- Electoral Register Server (ERS):
  prüft die Authentifizierung / Wahlberechtigung
- Validation Server (VS):
  entkoppelt die Stimmen der Wähler von den Wählern;
  prüft die Wahlberechtigung ebenfalls
- Ballot Box Server (BBS):
  verschlüsselt und speichert die Stimmen
- Tallying Component (TC):
  erhält die Stimmen offline[5] zählt diese Stimmen aus
- Comitee-Tool:
  startet und beendet die Wahl.
  Zum Starten und Beenden werden k Administratoren benötigt. Dieser Server kommuniziert den Status den anderen.

6.4.2 Wahlablauf

6.4.2.1 Pre-Election

- Die signierte Polyas-Software wird auf den Wahlserver installiert. Dass die unveränderte Software verwendet wird, ist essenziell für die Sicherheit, die das System bieten soll.
- Die TANs werden generiert und die folgenden Listen erzeugt:
  - Liste 1: (Wähler-ID, $h(TAN)$, h) $\rightarrow$ ERS
  - Liste 2: (Wähleradresse, $E(TAN)$) $\rightarrow$ Druckerservice
- Erzeugung der Schlüsselpaare für die Wahlserver.
  - HTTPS-Schlüsselpaar für den ERS, VS, BBS
  - Signaturschlüsselpaar für BBS
  - Schlüsselpaar für Datenverschlüsselung: Der öffentliche Schlüssel wird an den BBS geschickt, der geheime an die TC. Dabei ist der geheime Schlüssel mit zwei Passphrasen geschützt.
  - Die öffentlichen Schlüssel vom ERS und dem BBS werden veröffentlicht.
- Die folgenden Informationen werden an die Server verteilt und Tests werden durchgeführt.
  - Der ERS erhält das Wählerverzeichnis (Liste 1).
  - Der VS erhält die TANs im Klartext
  - Dass der Stimmenspeicher des BBS leer ist, wird getestet. Der Stimmenspeicher entspricht der Urne in papierbasierten Verfahren.
  - Der BBS erhält das Stimmzettellayout und die Wahlregeln.
- Die Zugriffsrechte für die Server werden festgelegt:
  - Pro Server werden zwei Accesstokens erstellt, von denen beide notwendig sind, um Zugriff zu erhalten.
  - Die sechs Accesstokens werden an die sechs verantwortlichen Personen im Election Committee verteilt.
  - Wie viele Accesstokens k für das CT notwendig sind, ist konfigurierbar. Diese werden ebenfalls in diesem Schritt erstellt und verteilt.
- Die Server werden konfiguriert: Installation von Firewalls, Anti-Viren-Software, etc
- Start der Wahl
  - An dem CT melden sich mindestens k Mitglieder des Election Committees an und
  - bestätigen alle den Startvorgang der Wahl.
  - Das CT spricht nun die einzelnen Server an um auf ihnen den Wahlvorgang zu starten.
  - Der Start des Wahlvorgangs muss von den jeweiligen Serveradministratoren separat bestätigt werden.
  - Bekanntgabe, dass die Wahl gestartet wurde.

[5] über einen externen Datenträger

6.4.2.2 Wahlphase

- Der Wähler registriert sich mit seiner ID und seiner TAN beim ERS
- der ERS prüft, ob $(ID, h(TAN))$ in der Liste enthalten ist (prüfen der Wahlberechtigung) und sendet, falls der Eintrag vorhanden ist, die TAN an den VS
- Der VS generiert, wenn die TAN in seiner Liste vorhanden ist, ein Wahltoken T
- – der VS sendet das Wahltoken T an den BBS. Wenn zu der TAN bereits ein Token erstellt wurde, wird dieser Schritt übersprungen.
  – der BBS speichert T und bestätigt den Erhalt beim VS
  – VS sendet T an den ERS
- – Der ERS speichert T und bestätigt den Erhalt beim VS
  – Der VS markiert die TAN als ungültig um mehrfache Stimmabgabe zu verhindern.
  – VS sendet eine Bestätigung, dass die TAN markiert wurde, an den ERS.
  – Wird erneut ein Token zu dieser TAN angefordert, wird das gleiche Token zurückgesendet.
- – Der ERS schickt T an den PC des Wählers
  – das VCI wird nun die Verbindung dem ERS trennen und eine Verbindung zum BBS aufbauen. Von nun an schickt das VCI bei jeder Nachricht, die es versendet das Token mit.
  – Das VCI fordert einen leeren Stimmzettel an.
  – Der BBS prüft das Token und antwortet, wenn das Token korrekt ist, mit einem leeren Stimmzettel.
  – der Wähler füllt den Stimmzettel aus
  – der Wähler schickt den Stimmzettel (V) im Klartext über eine SSL-verschlüsselte Verbindung an den BBS
  – Der BBS speichert den Stimmzettel hybrid verschlüsselt $(E_{pk_{TC}}(E_{key}(V),\ key))$ in der BallotBox ab.
  – Der BBS markiert den eben erstellten Eintrag als *selection*.
  – der BBS sendet V im Klartext über eine SSL-verschlüsselte Leitung an den Wähler zur Kontrolle.
  – Der Wähler fordert entweder einen neuen leeren Stimmzettel an, um seine Wahl zu korrigieren, oder er bestätigt die Stimmabgabe.
  – Wenn der Wähler die Stimmabgabe bestätigt, überprüft der BBS erneut das vom VCI mitgesendete Token.
  – Der BBS ändert den Status von *selection* auf *cast vote*.
  – BBS schickt das verwendete Wahltoken T an den ERS
  – BBS und ERS löschen T
- – Der ERS markiert die Wähler-ID mit *ungültig* bzw. *hat bereits gewählt*
  – Wähler erhält die Bestätigung der abgegebenen Stimme.

Der BBS fasst jeweils 30 Stimmen zu einem Block zusammen, hasht diesen zusammen mit dem Hashwert des vorherigen Blockes und sendet diesen an den ERS. Der BBS tut dies sobald 30 neue Stimmen abgegeben wurden.

6.4.2.3 Post-Election

- Wie beim Starten wird der Wahlvorgang durch k Mitglieder des Election Committees beendet.
  – Zuerst werden ERS und VS beendet. Das heißt, dass diese keinen neuen TANs mehr annehmen. Die Server werden jedoch noch nicht abgeschaltet.
  – Nach einer vorher definierten Zeitspanne Δt wird auch der BBS in einen Zustand versetzt, in dem er keine Stimmen mehr annimmt. (Δt ist im Originalpapier mit 10 Minuten angegeben.)
- Vorbereitung der Auszählung:
  – Die BallotBox wird auf ein externes Medium kopiert.
  – Dieses Medium wird zur TC gebracht und dort angeschlossen.
- Auszählung
  – Die Administratoren müssen ihre Passphrase eingeben, um den geheimen Schlüssel der TC zu entschlüsseln.
  – Die TC entschlüsselt die Stimmen und addiert alle Stimmen mit dem Status *cast vote*. Ein Mix ist nicht notwendig, da die Verbindung zum Wähler (TAN und Token) nicht auf dem Datenträger enthalten sind.

– Das Election Committee überprüft alle Vorgänge. Dazu gehören u.a. die Hashes der 30er Blöcke.
– Die Stimmen werden von der TC ausgedruckt, um die Wahl später unabhängig von einem Datenverlust prüfen zu können.
– Die Daten incl. aller vorhandenen Logs werden auf einer CD gespeichert.
– Das Wahlergebnis wird bekannt gegeben.

6.5 Common Criteria Protection Profile für Internetwahlen

6.5.1 Common Criteria

Die Common Criteria beschreibt Kriterien der Bewertung und Zertifizierung der Sicherheit von Computersystemen im Bezug auf Datensicherheit. Diese Schutzprofile (*Protection Profiles*) beschreiben die Sicherheitsanforderungen, die an ein System bestimmter Art gestellt werden, unabhängig von der Implementierung. Die Schutzprofile werden in Abhängigkeit der in ihr beschriebenen Anforderungen in sieben Gruppen eingeteilt:

- **EAL1** funktionell getestet
- **EAL2** strukturell getestet
- **EAL3** methodisch getestet und überprüft
- **EAL4** methodisch entwickelt, getestet und durchgesehen
- **EAL5** semiformal entworfen und getestet
- **EAL6** semiformal verifizierter Entwurf und getestet
- **EAL7** formal verifizierter Entwurf und getestet

Bis EAL4 wird eine Zertifizierung international gegenseitig anerkannt. Höhere Einstufungen müssen nicht anerkannt werden.

6.5.1.1 Aufbau eines Schutzprofils

Der Aufbau eines Schutzprofils orientiert sich an dem folgenden Schema:

1. Übersicht über die Sicherheitsanforderungen
   Informelle Beschreibung der formalen Definitionen, die später folgen
2. Notationen und Begriffsdefinitionen
3. Postulate zur Übereinstimmung
   auf welcher der CC-Version das Dokument beruht, welches EAL-Level es beschreibt, ...
4. Definition des Sicherheitsproblems
   - zu schützende Ziele
   - Angreifermodell (vom EAL-Level abhängig)
   - Bedrohungen
   - organisatorische policies
   - Annahmen über Gebrauch und Umgebung
5. Sicherheitsziele
6. IT-Sicherheitsanforderungen
7. Anforderungen an die Vertrauenswürdigkeit des Systems

6.5.1.2 Schutzprofil für Basissatz von Sicherheitsanforderungen an Online-Wahlprodukte

Das Dokument *Common Criteria – Schutzprofil für Basissatz von Sicherheitsanforderungen an Online-Wahlprodukte* [BSI08] beschreibt lediglich einen Basisschutz. Es ist nach EAL-Level 2 eingestuft. Der Fokus liegt dabei auf dem Schutz des Wahlgeheimnis und der Quittungsfreiheit, der Korrektheit und der Gleichheit bei der Wahl. Es ist keine Verifikation vorgesehen und fordert keinen Schutz gegen Erpressung gemäß des Sicherheitsbegriffes *coercion resistant*. Eine hohe Benutzerfreundlichkeit des Wahlsystems ist ebenfalls nicht Bestandteil der beschriebenen Anforderungen. Am Rande werden mögliche Stimmenkäufe und Erpressungen behandelt. Durch die gestellten Anforderungen ist Revoting jedoch ausgeschlossen. Für Systeme, die es dem Wähler ermöglichen die bereits abgegebene Stimme zu überschreiben, ist eine Anpassung der Anforderungen nötig. Das betrachtete Angreifermodell ist relativ schwach. Es werden insbesondere Laien und *normale Benutzer* als Bedrohung beschrieben.

Inhalt des Schutzprofils Die Wahl wird in drei Phasen unterteilt:

1. Vorbereitungsphase,
2. Wahl- und Auszählungsphase
3. Archivierungsphase

In dem Schutzprofil wird jedoch nur die Wahl und Auszählungsphase betrachtet. Die Vorbereitungsphase und die Archivierung sind nicht Gegenstand des evaluierten Objektes.

Das gesamte Schutzprofil zu beschreiben würde den Rahmen des Skriptes sprengen. Lediglich einige Besonderheiten werden dargestellt. (Für die Prüfung empfehlen wir das Dokument [BSI08] auszugsweise selbst zu lesen.)

Subjekte In der Definition des Sicherheitsproblems werden unter anderem die zu schützenden Werte (101) und die mit dem Protokoll interagierenden Subjekte (103, 104) aufgezählt. Als Angreifer (104) werden der Netzwerkangreifer, der registrierte Wähler, der unbefugte Wähler und Personen die nach der Stimmauszählung Zugriff auf die gespeicherten Daten Zugriff haben genannt. Der Wahlvorstand, der unter Umständen natürlich Teil der eben genannten Gruppen sein kann, wird allerdings nicht explizit aufgeführt. Dass es sich bei dem Wahlvorstand nicht um eine einzelne Person handeln kann, wird beispielsweise in 140 klar. Dort wird gefordert, dass keine Person alleine Zugang oder Zugriff zu den Servern des Systems haben darf.

Ablauf der Wahl Ein fester Ablauf der Wahl wird durch das Schutzprofil nicht vorgegeben. Es werden lediglich zwei Vorschläge gemacht. Sie unterscheiden sich in der Reihenfolge der Authentisierung und der Wahlentscheidung. In einem Ansatz kann jeder einen Stimmzettel anfordern und ausfüllen, ihn jedoch nur nach der erfolgreichen Authentisierung speichern lassen. Der andere Ansatz sieht vor, dass nur registrierte Wähler einen Wahlzettel erhalten können. Die Korrektur des Stimmzettels vor der endgültigen Abgabe ist in beiden Varianten vorgesehen.

Die Notwendigkeit beide Vorgehen vorzusehen ergibt sich aus der Unabhängigkeit des Schutzprofils von einer Implementierung. Je nach Protokoll kann es für die Prüfung der korrekten Arbeitsweise der Systeme erforderlich sein, dass unabhängige Instanzen die Funktion der Server testen können. Um die Glaubwürdigkeit dieser Art der Kontrolle zu erhöhen, muss es auch nicht registrierten Nutzern möglich sein, mögliches Fehlverhalten festzustellen und zu beweisen. Am Beispiel von Helios wird deutlich, dass eine Kontrolle wichtig sein kann, da der Server, der die Stimme des Nutzers verschlüsselt, jede beliebige statt der gewünschten Stimme verschlüsseln und abgeben könnte, wenn keine Tests möglich wären. Durch die Möglichkeit der Kontrolle, kann der Server bei einem solchen Verhalten ertappt werden.

Rückmeldung Der registrierte Wähler erhält eine zutreffende Rückmeldung über die Erlaubnis bzw. Verweigerung und den Erfolg bzw. Misserfolg seiner Stimmabgabe. (129)

Ohne das jedes Wort dieser Anforderung auf die Goldwaage legen zu wollen, kann die Aussage auf verschiedene Arten verstanden werden. Die naheliegende ist, dass der Wähler eine klare Aussage in Form einer Bildschirmausgabe darüber erhält, dass seine Stimme erfolgreich gespeichert wurde. Ob diese Aussage für einen potentiellen Angreifer noch eine Aussagekraft enthält, geht aus dieser Formulierung nicht klar hervor.

Für verschiedene Ansätze, die verhindern sollen, dass der Wähler erpresst werden kann, stellt das eine unüberwindbare Hürde dar. Die Konzepte von Fake-Keys oder Panikpasswörtern ist nicht mehr direkt anwendbar. Bei beiden Systemen steht dem Wähler die Möglichkeit offen, sich auf eine Art und Weise zu authentisieren, dass das System zwar keine Stimme annimmt, sich dem Nutzer gegenüber aber ununterscheidbar von der herkömmlichen Authentisierung verhält. Ununterscheidbar bedeutet in diesem Kontext aber auch, dass der Angreifer anhand der Meldungen, die der Nutzer erhält, nicht zwischen einer erfolgreichen und einer vorgetäuscht erfolgreichen Stimmabgabe unterscheiden kann.

Serverraum Das Schutzprofil fordert (145), dass während der Wahl nur der Wahlvorstand den Serverraum betreten darf. Das hat allerdings zur Folge, dass die Verwendung von kommerziellen Rechenzentren im Allgemeinen ein Problem darstellen wird. Für politische Wahlen wird es vermutlich kein Problem sein, eine entsprechende Infrastruktur zu schaffen. Für Vereine, Hochschulen und Bildungs- und Forschungseinrichtungen ist die Herausforderung jedoch größer.

Kapitel 7

Alternative Wahlformen und Sonderfälle

Bisher gingen wir immer davon aus, dass eine Wahl daraus besteht, dass der Wähler aus einer Menge an Auswahlmöglichkeiten (Kandidaten) eine auswählt, oder, falls mehrere Stimmen zu vergeben sind, diese gleichgewichtig auf die Kandidaten verteilt werden. Dies ist aber nicht immer der Fall, so werden z.B. in Australien (und auch woanders) parlamentarische Wahlen über eine sogenannte Präferenzwahl (Abschnitt 7.1) durchgeführt. Auch ist es bei einigen Wahlen die Vorschrift, dass der Wähler die Möglichkeit haben muss, eine Stimme für einen Kandidaten abzugeben, der nicht auf der Kandidatenliste steht, einen sogenannten *write-in-Kandidaten*, mehr dazu in Abschnitt 7.2. Eine weitere alternative Wahlform ist das *Delegated Voting* zur Umsetzung einer *Liquid Democracy*, hier hat der Wähler die Möglichkeit, seine Stimmabgabe an einen anderen Wähler zu delegieren. Hiermit befasst sich Kapitel 7.3.

7.1 Präferenzwahl

Bei einer *Präferenzwahl* (STV von engl. *single transferable vote*, im deutschen auch als Übertragbare Einzelstimmgebung bezeichnet) besteht eine Stimmabgabe nicht aus der Auswahl eines (oder mehrerer) Kandidaten, sondern den zur Wahl stehenden Kandidaten wird vom Wähler je eine Priorität oder eine Präferenz zugewiesen. Der Wähler bringt die Kandidaten auf dem Stimmzettel also in eine Reihenfolge oder eine Rangfolge. Die erste Präferenz ist dabei der bevorzugte Kandidat. Es gibt verschiedene Formen der Präferenzwahl, in einigen Wahlen wird eine vollständige Priorisierung auf jedem (gültigen) Stimmzettel vorgeschrieben. In anderen Wahlen kann der Wähler auch nur die ersten paar Präferenzen angeben und die anderen frei lassen. Das Wahlverfahren Shuffle Sum (Abschnitt 7.1.2), das wir hier betrachten, geht davon aus, dass der Wähler eine vollständige Priorisierung der Kandidaten angibt. Jeder Stimmzettel impliziert damit eine Permutation der Kandidaten, enthält also deutlich mehr Information als die Auswahl eines einzelnen Kandidaten. Wärend es für Stimmzettel mit nur einer Auswahl genügt, die Verbindung zwischen Wähler und Stimmzettel geheim zu halten, reicht dies bei der Präferenzwahl nicht aus, denn der Angreifer könnte den Wähler zwingen, seinem bevorzugten Kandidaten die erste Präferenz zuzuweisen und in den Rest der Permutation seine Idendität zu codieren. Weitere Angriffe werden in [BMN+] beschrieben. Wie man sieht hat man hier eine noch schwieriger zu überwindende Kluft zwischen der Verifizierbarkeit und der nicht-Erpressbarkeit.

Präferenzwahlen werden häufig in Szenarien angewendet, in denen mehrere Kandidaten auf mehrere zur Verfügung stehende Sitze verteilt werden, es wird also nicht nur ein Kandidat ausgewählt. Die Idee der Präferenzwahl ist nun, dass möglichst wenig Stimmgewicht "verloren gehen" soll: Bekommt die erste Präferenz eines Wählers zu wenig Stimmen für einen Sitz, so schafft es vielleicht die zweite, der Wähler ist also bei der weiteren Auswahl trotzdem nicht außen vor. So kann er seine erste Präferenz auch einem Kandidaten geben, von dem man weiß, dass er wahrscheinlich wenig Chancen auf einen Sitz hat, und hat doch die Sitzverteilung effektiv mitbestimmt, indem er die zweite Präferenz einem wahrscheinlicheren Kandidaten gibt.

Vorsicht: Die Präferenzwahl ist eng verwandt mit aber nicht das selbe wie die *Rangfolgewahl*, die in dieser Vorlesung und diesem Skript nicht behandelt wurde. Bisweilen werden die Bezeichnungen auch versehentlich redundant verwendet, falls mir das in der Vorlesung passiert ist, Entschuldigung :-)

Sowohl die Präferenzwahl als auch die Rangfolgewahl bringen einige interessante, in der Vorlesung nicht behandelte Anomalien mit sich. So könnte es unter Umständen taktisch geschickt sein, seine erste Präferenz einem unbeliebten Kandidaten zu geben und den bevorzugten Kandidaten auf die zweite Präferenz zu legen. Dem interessierten und noch nicht so sehr im Lernstress befindlichen Leser wird hier etwas Stöbern auf Wikipedia und dort verwiesener Literatur empfohlen.

7.1.1 Auszählung einer Präferenzwahl

Wir beschreiben hier die Version der Präferenzwahl, in der der Wähler eine vollständige Permutation der Kandidaten angibt. Dabei sei n die Anzahl Wähler, l die Anzahl Kandidaten und s die Anzahl zu vergebender (gleichwertiger) Sitze. Die Auszählung in einer Präferenzwahl erfolgt in mehreren Runden. Als weiteren Parameter brauchen wir dabei eine *Quote* q. Ein Kandidat zählt als gewählt, wenn er in einer durchgeführten Runde mindestens so viele Stimmen wie die Quote q erhält.

Hierbei wird oft die *Droop Quota* verwendet:

Droop Quota Die Droop Quota berechnet sich mit

$$q := \left\lfloor \frac{n}{s+1} \right\rfloor + 1$$

und bestimmt die kleinste Quote, so dass in keiner Runde mehr Kandidaten gewählt werden (also die Quote erreichen), als noch Sitze zu vergeben sind.

Die Auszählung erfolgt nun in mehreren Runden, in denen je folgende Schritte durchgeführt werden, dabei hat in der ersten Runde jeder Stimmzettel das Gewicht eins, die Stimmzettel werden jedoch im Verlauf weiterer Runden teilweise umgewichtet.

1. **Zähle erste Präferenz aus:** Summiere für jeden Kandidaten das Gewicht der Stimmzettel, die diesen Kandidaten als erste Präferenz haben[1]. Wir nennen die hier berechneten Summen die Anzahl Stimmen der einzelen Kandidaten in der aktuellen Runde, man beachte aber, dass die Summen ab der zweiten Runde durch die Umgewichtung im Allgemeinen keine ganzen Zahlen mehr sind.

2. **Bestimme *gewählte* oder *zu löschenden* Kandidaten:**
   - **Wählen:** Jeder Kandidat, der in Schritt 1 der aktuellen Runde die Quote q erreicht hat, zählt als gewählt.
   - **Löschen:** Wenn in der aktuellen Runde kein Kandidat die Quote erreicht, wird der Kandidat mit den wenigsten Stimmen in Schritt 1 der aktuellen Runde (also den wenigsten ersten Präferenzen) als *gelöscht* markiert.

3. **Passe Stimmzettelgewicht an:** Für jeden in Schritt 2 der aktuellen Runde gewählten Kandidaten C werden alle Stimmzettel, auf denen dieser Kandidat die erste Präferenz hatte, umgewichtet. Dazu wird zuerst für jeden gewählten Kandidaten C der sogenannte *Transfer Value* t_C dieses Kandidaten berechnet. Sei m_C die im ersten Schritt der aktuellen Runde erlangte Anzahl Stimmen von C. Dann berechnet sich sein Transfer Value zu

   $$t_C := \frac{m_C - q}{m_C}.$$

   Für jeden Stimmzettel, auf denen ein gewählter Kandidat C erste Präferenz war, wird nun das neue Gewicht berechnet: Sei w_{alt} das bisherige Gewicht des Stimmzettels. Dann beträgt das neue Gewicht $w := w_{alt} * t_C$. Wie man sieht, beträgt der Transfer Value eines Kandidaten, der genau die Quote erreicht, $t_C = 0$. Entsprechend gibt es kein überschüssiges Stimmgewicht, die Stimme der entsprechenden Stimmzettel wurde also vollständig aufgebraucht, und das neue Gewicht des Stimmzettels ist $w = 0$. Erhält ein Kandidat mehr Stimmen als die Quote, so wird das überschüssige Stimmgewicht verteilt auf die Stimmzettel, die diesen Kandidaten als erste Präferenaz hatten: Deren Stimme wurde "nicht ganz aufgebraucht". Der Transfer Value ist also sozusagen das überschüssige Stimmgewicht, das jedem, der diesen Kandidaten C gewählt hat, noch zusteht, und daher auf die jeweilige nächste Präferenz der Stimmzettel übertragen wird.

[1] Würde nur eine Runde mit nur diesem Schritt durchgeführt werden, so entspräche dies einer normalen Mehrheitswahl: Die erste Präferenz entspricht unserem Kreuzchen. Nach der ersten Runde sieht man also, wer bei der Mehrheitswahl gewonnen hätte.

4. **Entferne Kandidaten:** Alle in Schritt 2 der aktuellen Runde gewählten Kandidaten, bzw. der als gelöscht markierte, falls keiner gewählt wurde, werden für alle weiteren Runden von allen Stimmzetteln entfernt. Die Präferenzen aller Stimmzettel werden angepasst, indem jeweils alle Kandidaten mit schlechterer Präferenz als der entfernte Kandidat aufrücken. Haben die Kandidaten (A, B, C) z.B. auf einem Stimmzettel die Präferenzen $(A : 1, B : 2, C : 3)$ und Kandidat A wird entfernt, so hat der neue Stimmzettel nun die Präferenzen $(B : 1, C : 2)$. Bei einem Stimmzettel mit Präferenzen $(C : 1, A : 2, B : 3)$ würde das Entfernen von A die Präferenzen $(C : 1, B : 2)$ ergeben.

Diese vier Schritte werden nun durchgeführt, bis alle zur Verfügung stehenden Sitze belegt sind, oder manchmal auch, bis nur noch so viele Kandidaten wie Sitze übrig sind, die verbleibenden Kandidaten bekommen dann die übrigen Sitze. Wie gesagt, wird die Droop-Quota verwendet, so werden in jeder Runde nur so viele Kandidaten gewählt, wie noch Sitze frei sind.

Das Umgewichten wird manchmal auch so beschrieben, dass für jeden gelöschten Kandidaten ein Transfer Value berechnet wird. Entfernte, nicht-gewählte Kandidaten haben dabei Transfer Value $t = 1$, die gewählten wie gehabt. Das Ergebnis ist dasselbe.

7.1.2 Das Wahlverfahren Shuffle Sum

Shuffle Sum [BMN+] ist ein Verfahren zur Auszählung einer Präferenzwahl, das keine einzelen Stimmzettel offenlegt, jedoch trotzdem die Auszählung nachvollziehbar durchführt. Dazu werden von Shuffle Sum vier verschiedene Darstellungen des Stimmzettels verwendet, zwischen denen per verifizierbarem Mix gewechselt werden kann, und die nie den vollständigen Stimmzettel oder eine Information über die Auswahl des Wählers offenlegen.

Da sich Shuffle Sum nur um die Auszählung selbst kümmert, muss eine sichere Stimmabgabe als gegeben vorausgesetzt werden.

7.1.2.1 Voraussetzungen

Es sei im Folgenden n die Anzahl Wähler, l die Anzahl Kandidaten, s die Anzahl zu vergebender Sitze, und E die verwendete Verschlüsselungsfunktion.

Für die Verschlüsselungsfunktion E müssen folgende Voraussetzungen gelten:

- Die Verschlüsselungsfunktion E ist additiv homomorph: Es gibt eine Operation $\oplus$ sodass für alle Nachrichten m_1, m_2 im Klartextraum gilt:

  $$E(m_1) \oplus E(m_2) = E(m_1 + m_2),$$

  die Addition[2] kann also auf den Chiffraten durchgeführt werden. Wir schreiben $\sum_i E(m_i)$ für $E(m_1) \oplus E(m_2) \oplus \ldots$, auch wenn es sich bei der Operation $\oplus$ selbst nicht um eine Addition handeln sollte. Wir symbolisieren damit die Berechnung der Addition der Klartexte auf den Chiffraten unter Ausnutzung der Homomorphieeigenschaft.

- Die Verschlüsselung E erlaubt *Threshold-Entschlüsselung* (siehe Abschnitt 2.4.4.5).

- Die Verschlüsselung E erlaubt Wiederverschlüsselung[3].

Geeignet als Verschlüsselungsfunktion E wäre hier u.A. das exponentielle ElGamal zusammen mit der ElGamal-Threshold-Entschlüsselung oder das in dieser Vorlesung nicht behandelte Pallier-Verfahren.

Voraussetzungen für die Stimmabgabe:

- Die Stimmabgabe erfolgt sicher und in einer Weise, dass der Wähler seine Stimmabgabe prüfen kann, ohne erpressbar zu sein.

- Der Inhalt der Stimmezettel ist in geeigneter Weise[4] teilweise mit E verschlüsselt.

[2] Nicht irritieren lassen: Bei einigen Verfahren wie z.B. exponentielles ElGamal ist $\oplus$ die Multiplikation: Man muss Chiffrate multiplizieren um die beinhalteten Klartexte zu addieren.

[3] Dieser Punkt ist redundant, die Wiederverschlüsselbarkeit folgt hier aus der Homomorphieeigenschaft sowieso.

[4] Auf das genaue Format der abgegebenen Stimmzettel wird später noch eingegangen.

7.1.2.2 Die Stimmzetteldarstellungen

Es gibt vier verschiedene Darstellungen der Stimmzettel, zwischen denen wärend der Auszählung per Mix und Wiederverschlüsselung gewechselt wird. Jede Darstellung enthält alle Information, die die (evtl. aus mehreren Parteien bestehende) Wahlauthority benötigt, um einen Stimmzettel in jede der anderen Darstellungen transformieren zu können. So enthält jede Darstellung ein Chiffrat des aktuellen Stimmzettelgewichts sowie entweder die verschlüsselten Kandidaten in Präferenzreihenfolge oder die verschlüsselten Präferenzen in Kandidatenreihenfolge. Dabei meinen wir mit Kandidatenreihenfolge die für jeden Stimmzettel identische, vor der Wahl festgelegte Reihenfolge, in der die Kandidaten auf dem Stimmzettel stehen. Die Präferenzreihenfolge spiegelt die Auswahl des Wählers wider: Der Kandidat mit des Wählers erster Präferenz steht vorne, dann die zweite Präferenz u.s.w. Die Darstellungen wären:

Kandidatenreihenfolge-Darstellung Der Stimmzettel besteht aus einer Liste verschlüsselter Präferenzen in Kandidatenreihenfolge, sowie einem Chiffrat des aktuellen Stimmzettelgewichts.

Präferenzreihenfolge-Darstellung Der Stimmzettel besteht aus einer Liste verschlüsselter Kandidaten in Präferenzreihenfolge, sowie einem Chiffrat des aktuellen Stimmzettelgewichts.

entfernte-Kandidaten-Darstellung Stimmzettel in Präferenzreihenfolge-Darstellung, mit zusätzlichem verschlüsselten *gelöscht-Flag* hinter jedem Kandidaten. Das Flag ist 1, falls der entsprechende Kandidat am Ende der aktuellen Runde entfernt wird (weil er gewählt oder gelöscht wurde), und 0 sonst. Das aktuelle Stimmzettelgewicht steht auch hier verschlüsselt auf dem Stimmzettel.

Gewichts-Darstellung Im Originalpapier wird diese Darstellung *First Preference Ballot* genannt. Wir nennen es hier die *Gewichts-Darstellung*, weil hier das Gewicht des Stimmzettels die Hauptrolle spielt. Diese Darstellung wird benutzt, um die erste Präferenz auszuzählen und um die Umgewichtung der Stimmzettel durchzuführen. In dieser Darstellung besteht der Stimmzettel aus einer Liste von verschlüsselten vermeintlichen "Gewichten" in Kandidatenreihenfolge, dabei hat die aktuelle erste Präferenz des Stimmzettels das Gewicht des Stimmzettels, und alle anderen das Gewicht 0. Auch die Präferenzen stehen verschlüsselt in Kandidatenreihenfolge dabei, aber nur, um den Stimmzettel später wieder in die anderen Darstellungen überführen zu können.

Wie die Stimmzetteldarstellungen ineinander überführt werden können, bleibt dem Leser als Übung überlassen[5].

7.1.2.3 Auszählung

Wir gehen nun davon aus, dass alle Stimmzettel in der Kandidatenreihenfolge-Darstellungen vorliegen (Präferenzreihenfolge geht auch). Alle Stimmzettel haben in der ersten Runde das Gewicht 1, das Gewicht muss also am Anfang nicht geheimgehalten werden. Erst später könnte man prinzipiell über die Gewichtsveränderungen Information über die Auswahl des Wählers erlangen. Die Wahlauthority kann nun zu jedem Stimmzettel ein Chiffrat mit dem Gewicht 1 hinzufügen und beweisen, dass das Chiffrat die 1 enthält. Außerdem wird die Quote q, die ein Kandidat erreichen muss, um gewählt zu werden, wie oben beschrieben festgelegt und berechnet. Die Auszählung erfolgt dann in mehreren Runden, in denen immer wieder die oben beschriebenen vier Schritte durchgeführt werden. (Hier hat sich noch ein fünfter als Abbruchkriterium mit eingeschlichen.) Wie genau Shuffle Sum diese vier Schritte durchführt, wird nun beschrieben:

1. **Erste Präferenz auszählen:** Transformiere alle Stimmzettel in die Gewichtsdarstellung. Nun wird für jeden Kandidaten von jedem Stimmzettel das Gewicht, das bei diesem Kandidaten steht, aufsummiert Zur Erinnerung: nur bei der jeweils ersten Präferenz steht ein Gewicht ungleich 0. D.h. es wird für jeden Kandidaten das Gewicht der Stimmzettel aufsummiert, die diesen Kandidaten als erste Präferenz haben, genauso geht von jedem Stimmzettel auch nur in eine der Summen, nämlich die des Kandidaten mit der ersten Präferenz, ein Wert ungleich 0 ein. Da das verwendete Verschlüsselungsverfahren additiv homomorph ist, kann die Summe gebildet werden, indem die Addition auf den Chiffraten durchgeführt wird,

[5] Man kann auch in [BMN+] nachsehen.

und die Summe dann via Threshold-Decryption (siehe Abschnitt 2.4.4.5) von den Mitgliedern der Wahlauthoriy entschlüsselt wird, selbverständlich mit Beweis für korrektes Entschlüsseln. Diese Rechenschritte können und sollten alle öffentlich geschehen.

2. **Kandidaten wählen oder löschen:** Nun wird geprüft, welche der in Schritt 1 berechneten Summen die Quote erreichen oder übersteigen. Die entsprechenden Kandidaten zählen als gewählt. Falls in einer Runde kein Kandidat die Quote erreicht, gilt der Kandidat mit den wenigsten Stimmen (also der mit der kleinsten Summe in Schritt 1 der aktuellen Runde) als gelöscht.

3. **Stimmzettelgewicht anpassen:** Das ist der komplizierteste Schritt. Falls kein Kandidat gewählt wird, ist hier nichts zu tun, weiter mit Schritt 4. Ansonsten: Sei $S = \{C_1, \ldots, C_k\}$ die Menge der gewählten Kandidaten, t_i jeweils die in Schritt 1 der aktuellen Runde berechnete Anzahl Stimmen von Kandidat C_i. Berechne nun für jeden Kandidaten in S den Transfer-Value $s_i = 1 - \frac{q}{t_i}$. Das Umgewichten wird nun etwas anders durchgeführt als oben beschrieben: Da es schwer ist, auf Chiffraten mit verschlüsselten nicht-ganzzahligen Werten zu multiplizieren, wird dies vermieden, indem der Transfer-Value angenähert wird durch Werte a_i, d_i mit $s_i \approx \frac{a_i}{d_i}$. Die Idee ist nun, die Stimmzettelgewichte nicht durch Multiplikation mit s_i zu reduzieren, sondern die betroffenen Stimmzettel[6] "nur" mit a_i zu multiplizieren und alle anderen mit d_i. Um das ganze etwas vernünftiger zu machen, werden die Annäherungen zuerst auf einen gemeinsamen Nenner $d = kgV(\{d_i : C_i \in S\})$ gebracht (kgV ist das kleinste gemeinsame Vielfache), dadurch erhält man neue Zähler a_i', die Stimmzettelgewichte mit einem Kandidaten $C_i \in S$ als erste Präferenz werden mit dem entsprechenden a_i' multipliziert und alle übrigen mit d. Durchgeführt wird die Gewichtsanpassung auf den Stimmzetteln in ihrer Gewichts-Darstellung, schliesslich sieht man den Stimmzetteln in keiner Darstellung an, welcher Kandidat die erste Präferenz hat, entsprechend weiß man in einer anderen Darstellung auch nicht, welches Gewicht man mit a_i' multiplizieren muss und welches mit d. Nun wird auf jedem Stimmzettel für jeden Kandidaten $C_i \in S$ das entsprechende vermeintliche Gewichtschiffrat mit $a_i' := a_i \frac{d}{d_i}$ multipliziert. Für alle übrigen Kandidaten wird das entsprechende Gewichtschiffrat mit d multipliziert. Wir erinnern uns hier nochmal, dass in dieser Darstellung bei jedem Kandidaten ein verschlüsseltes Gewicht steht, nur eins davon ist ungleich 0, entsprechend wird nur eins davon, nämlich das Stimmzettelgewicht, angepasst, und zwar auf die richtige Art und Weise. Es wird dem Leser hier empfohlen, sich nochmal klar zu machen, warum das so ist.

4. **Kandidaten entfernen:** Die gewählten Kandidaten bzw. der gelöschte Kandidat, falls in der aktuellen Runde keiner gewählt wurde, werden nun entfernt: Dazu werden zuerst die Präferenzen angepasst, und danach die Kandidaten gelöscht. Um die Präferenzen anzupassen, wird zuerst jeder Stimmzettel in die entfernte-Kandidaten-Darstellung transformiert. Danach passiert für alle Stimmzettel folgendes: Für alle Kandidaten $C_i, i = 1, \ldots, l$ sei p_i die Präferenz von C_i auf dem aktuellen Stimmzettel und x_i das Flag, das bestimmt, ob C_i gelöscht wird[7]. Auf dem Stimmzettel liegen diese Werte mit E verschlüsselt vor, die Zuordnung der $E(p_i)$ zu $E(x_i)$ ist jedoch bekannt (sie stehen hintereinander). Für jeden Kandidaten $C_i, i = 1, \ldots, l$ wird nun die neue Präferenz folgendermaßen berechnet:

$$E(p_i^{neu}) := E(p_i) - \sum_{j=1}^{i} (E(x_j)),$$

in Worten, für jeden Kandidaten, der niedrigere Präferenz als C_l hat und in der aktuellen Runde gelöscht wird, wird die Präferenz von C_l um eins verringert, um aufzurücken. Nach der Präferenzanpassung wird der Stimmzettel in die Kandidatenreihenfolge-Darstellung gebracht und alle gelöschten Kandidaten entfernt.

5. **Abbruchkriterium:** Sind am Ende einer Runde nur noch so viele Kandidaten wie freie Sitze übrig, so zählen diese restlichen Kandidaten als gewählt.

[6] Also die mit Kandidaten C_i als erste Präferenz

[7] Das Flag kann öffentlich berechnet werden: welche Kandidaten gelöscht werden, ist öffentlich bekannt. Es kann also in einer Darstellung in Kandidatenreihenfolge den Kandidaten zugeordnet werden. Bei der Überführung in die entferne-Kandidaten-Darstellung muss es jedoch zusammen mit dem Chiffrat des zugehörigen Kandidaten durch Mischen und Wiederverschlüsseln verschleiert werden, so dass man nicht am Chiffrat des Flags erkennt, welcher Kandidat welche Präferenz hat.

7.1.2.4 Sicherheit des Verfahrens

Das Wahlgeheimnis ist höchstens so sicher wie die Verschlüsselungsfunktion[8] E. Inwieweit weitere in der Auszählung bekanntwerdende Daten Erpressbarkeit ermöglichen, wird in [BMN+] diskutiert. Mit Ranking-Wahlen ist das Thema etwas schwieriger, da die Information, welche Kandidaten in einer Runde gewählt oder gelöscht werden, oder auch die Nachkommastellen der in den einzelnen Runden berechneten Summen, aus denen man wiederum auf bestimmte Gewichte schließen kann, Teilinformation über die An-oder Abwesenheit bestimmter Permutationen auf Stimmzetteln enthalten könnten. Dieses komplexe Thema wird hier nicht weiter behandelt, es sei hier nur auf das Problem hingewiesen, dem interessierten Leser wird empfohlen, sich [BMN+] sowie weiterführende Literatur anzuschauen.

7.2 Write-In-Kandidaten

Quellen: [Kem12, CCC+10] In einigen Regionen, wie z.B. Takoma Park, USA, hat der Wähler per gesetzlicher Vorschrift die Möglichkeit, eine Stimme für einen Kandidaten abzugeben, der nicht auf dem Stimmzettel bzw. der Kandidatenliste steht. Bei einer Papierwahl enthält der Stimmzettel hierfür einen vorgesehenen Bereich, in den der Wähler seinen Wunschkandidaten eintragen kann, man spricht daher von *Write-in-Kandidaten.*

TODO

7.3 Delegated Voting/ Liquid Democracy

Eine weitere aus kryptographischer Sicht sehr interesannte Wahlform ist das Delegated Voting zur Umsetzung einer Liquid Democracy. Sie kann gesehen werden als eine Zwischenstufe zwischen einer direkten Demokratie und einer repräsentativen Demokratie: Der Wähler kann sich entweder wie bei der direkten Demokratie an einem Entscheidungsfindungsprozess selbst beteiligen oder sein Stimmgewicht abgeben an einen Vertreter seines Vertrauens.

In Deutschland wird Delegated Voting unter anderem zur Entscheidungsfindung innerhalb der Piratenpartei eingesetzt. Delegated Voting kann wie die anderen Wahlformen auch trotz der politischen Motivation auch für nicht-politische Wahlen interessant sein, z.B. zur Entscheidungsfindung in Vereinen.

7.3.1 Delegated Voting in a Nutshell

Über die Prinzipien der Liquid Democracy bzw. des Delegated Voting gibt es ganze Papiere [For02, GA10] und weder eine einheitliche Definition noch einheitliche Namen, in-a-Nutshell beschrieben kann sie gesehen werden als eine Zwischenstufe zwischen einer direkten Demokratie und einer repräsentativen Demokratie. In der direkten Demokratie werden alle Entscheidungen direkt vom Volk getroffen (Bsp. Athen im 5. Jahrhundert), wer sich in den demokratischen Prozess einbringen möchte, muss sich jedoch persönlich an entsprechenden Abstimmungen beteiligen. Dabei hat jedoch eigentlich niemand ausreichend Kompetenz in jedem zu entscheidenden Thema oder auch die Zeit, sich ausreichend mit der Thematik zu befassen, sodass sich jeder nur an einem Teil der Entscheidungen sinnsvoll beteiligen kann. Als Gegenstück dazu delegiert das Volk in einer repräsentativen Demokratie sämtliche Entscheidungsmacht an Vertreter (z.B. unser Parlament), die dann allein für alle Entscheidungsfindungen innerhalb eines festgelegten Zeitraums zuständig sind.

Delegated Voting ist ein Kompromiss zwischen diesen beiden Demokratieformen: Beim Delegated Voting kann sich jedes Individuum zu jeder Zeit selbst entscheiden, ob es an einem Entscheidungsfindungsprozess persönlich beteiligt sein oder dies an einen Vertreter seines Vertrauens, einen sogenannten *Proxy*[9], delegieren möchte. Delegiert ein Wähler seine Stimme an einen Proxy, so erhält der Proxy das Stimmgewicht des Wählers, normalerweise für alle Entscheidungen in einem bestimmten Themengebiet bis ihm der Wähler das Stimmgewicht wieder entzieht, oder für die nächste Abstimmung, das ist Umsetzungssache. Dabei gelten folgende Bedingungen:

- Der Wähler kann sich bei jeder Entscheidungsfindung entscheiden, ob er selbst wählen, oder seine Stimme an einen Proxy seiner Wahl delegieren möchte.

[8] Also bei ElGamal höchstens so schwer zu brechen wie das DLog-Problem
[9] Delegated Voting wird daher in mancher Literatur auch mit Proxy Voting bezeichnet.

- Der Wähler kann sich vor Ende der Abstimmungsphase jederzeit umentscheiden und eine delegierte Stimmen dem Proxy wieder entziehen.
- Delegationen sind transitiv, d.h. ein Proxy kann das auf ihn fallende Stimmgewicht wiederum an einen weiteren Proxy delegieren[10].
- Jeder Wähler kann prinzipiell als Proxy agieren, wenn er möchte, bzw. die Voraussetzungen, Proxy zu werden, sollten leicht zu erfüllen sein.

7.3.1.1 Regeln nach Bryan Ford

Bryan Ford hat in [For02] die Anforderungen an eine Liquid Democracy[11] zusammengefasst:

- Choice of Role (Jeder kann sich aussuchen, ob er Proxy werden möchte oder nicht)
- Low Barrier to Participation (Es sollte leicht sein, Proxy zu werden, es darf aber eine kleine Hürde bestehen.)
- Delegated Authority (Ein Proxy bringt mit seiner Entscheidung sein eigenes Stimmgewicht ein sowie das Stimmgewicht der Wähler, die ihm ihre Stimme delegiert haben. Unterschiedliche Proxies stimmen entsprechend auch mit unterschiedlich hohem Stimmgewicht ab.)
- Privacy of Individuals (Für nicht-Proxies bleibt das Wahlgeheimnis gewahrt, dies bezieht sich sowohl auf die eigene Stimmabgabe als auch auf die eventuelle Auswahl des Proxies, ein Proxy weiss also nicht, welche Wähler ihr Stimmgewicht an ihn delegiert haben.)
- Accountability of Delegates (Proxies müssen für ihre Entscheidungen zur Rechenschaft gezogen werden können und sollten entsprechend offen abstimmen.)
- Specialization by Re-Delegation (Auch Proxies können ihr Stimmgewicht delegieren.)

Die genauen Auslegungen der einzelnen Anforderungen gehen hier in der Literatur und in verschiedenen Implementierungen des Delegated Voting auseinander. Auch werden nicht alle Anforderungen von jedem Verfahren, das delegated voting erlaubt, umgesetzt. So erlaubt Liquid Feedback z.B. keine Rollenwahl, jeder Wähler ist prinzipiell auch ein Proxy. Da in diesem System sowieso offen (bis auf den Einsatz von Pseudonymen) abgestimmt wird, widerspricht sich dies hier nicht mit dem Punkt Privacy of Individuals.

7.3.1.2 Eigenheiten des Delegated Voting

Bevor wir uns den kryptographischen Herausforderungen des Delegated Voting widmen, stellen wir hier noch der Vollständigkeit halber einige Eigenheiten des Delegated Voting vor.

Ringschluss Durch das transitive Delegieren verbunden mit dem Wahlgeheimnis kann es passieren, dass versehentlich ein Ringschluss entsteht: Proxy A delegiert an Proxy B, der an C delegiert, und C delegiert wiederum an A. So gibt keiner der Proxies eine Stimme ab und das gesamte an A, B und C delegierte Stimmgewicht geht verloren. In der Agora-Implementierung (Abschnitt 7.3.3) wird der Wähler auf solche Ringschlüsse hingewiesen und ggf. gewarnt. Eine ausführliche Beschreibung der Ringschluss-Auflösung finden sich auf der Agora-Webseite.

Unbekannte Verantwortung Je nach Umsetzung wissen die Proxies bis zum Ende der Wahl nicht, mit welchem Stimmgewicht sie abstimmen. Es könnte also prinzipiell passieren, dass ein Proxy eine Entscheidung allein trifft, ohne sich dessen bei der Abstimmung bewusst zu sein. Deshalb sollte jeder Proxy mit seiner Entscheidung verantwortungsvoll umgehen.

Ungewollte Transitivität Angenommen, ein Wähler W vertraut Proxy A, jedoch nicht Proxy B. Durch die Transitivität der Delegation kann A an B delegieren, wodurch das Stimmgewicht von W an B übergeht, obwohl W das vielleicht nicht wollte. Bei einer offenen Wahl der Proxys wird W dies mitbekommen und kann sich dann entscheiden, wie er bei weiteren Abstimmungen mit diesem Wissen umgeht.

[10] In der Literatur wird wohl davon ausgegangen, dass das Stimmgewicht eines Proxies immer als ganzes weitergegeben wird, ein Aufteilen des Stimmgewichts an mehrere Proxies wurde unseres Wissens nirgends erwähnt.
[11] Ford nennt sie *Delegative Democracy*.

7.3.1.3 Kryptographische Herausforderungen

Das Delegated Voting stellt uns Kryptographen vor eine ganze Menge neuer Herausforderungen, einige davon seien hier genannt:

- Verifizierbarkeit und Nicht-Erpressbarkeit stehen hier noch mehr im Widerspruch als bei herkömmlichen Wahlen, denn nicht nur die Stimme soll geheimgehalten werden, sondern auch die Tatsache, ob und an wen ein Wähler seine Stimme delegiert hat.
- Gleichzeitig muss das Stimmgewicht der Proxies korrekt und nachvollziehbar berechnet werden können, und der Wähler, oder auch ein delegierender Proxy, sollte nachprüfen können, dass sein Stimmgewicht an den richtigen Proxy übergegangen ist.
- Außerdem sollte der Wähler nachprüfen können, dass sein Stimmgewicht einem Proxy ggf. auch wieder entzogen wurde.
- Die Möglichkeit, sich jederzeit umzuentscheiden, verlangt nach einer sicheren Umsetzung von Revoting, dessen Probleme bereits in Abschnitt 6.3.1 beschrieben wurden. In Agora wird das Problem in gewisser Weise umgangen, indem die Proxies vor der eigentlichen Wahlphase ihre Stimme abgeben. Wähler entscheiden somit erst nach der Stimmabgabe der Proxies, ob sie "das selbe tun wie Proxy X" oder nicht.
- Proxies werden hier natürlich durch ihr höheres Stimmgewicht auch zu einem beliebteren Ziel von Stimmkauf, Manipulation und Erpressung, gleichzeitig haben sie aber kein Wahlgeheimnis. Hier stellt sich die Frage, inwieweit man den Proxies ein gewisses Maß an Wahlgeheimnis gewähren kann oder sollte, ohne die Accountability ganz aufzuheben. So wäre es evtl. denkbar, dass ein Proxy per *designated verifier proof*[12] oder ähnlichem den Wählern gegenüber, die an ihn delegiert haben, seine Stimme offenlegen muss, aber sonst niemandem. Was hier politisch gesehen und kryptographisch gesehen die beste Lösung ist, soll hier als offene Frage stehenbleiben.

Eine Lösung für diese Probleme wird hier nicht angeboten. Es existiert momentan unseres Wissens kein kryptographisches Wahlverfahren, dass Delegated Voting mit allen beschriebenen Anforderungen sicher umsetzt. Im folgenden werden zwei Wahlverfahren vorgestellt, die Delegated Voting erlauben, jedoch noch nicht alle oben beschriebenen Probleme lösen.

7.3.2 Liquid Feedback

Liquid Feedback [liq] ist ein Open Source Projekt für online-Abstimmungen, und eins der ersten seiner Art, die eine Stimmdelegation vorsehen. Da es sich bei Liquid Feedback nicht um ein kryptogaphisches Wahlverfahren handelt, wird es hier nur kurz vorgestellt.

7.3.2.1 Ablauf einer Abstimmung

Hier soll ein kurzer Überblick über den Ablauf einer Abstimmung gegeben werden. Eine Abstimmung durchläuft in Liquid Feedback mehrere Phasen:

Neu-Phase In der Neu-Phase werden ein- oder mehrere Anträge zu einem Thema erstellt. Die Benutzer des Liquid-Feedback-Systems können ihr Interesse kundgeben.

Diskussion Haben genügend Benutzer in der Neu-Phase interesse gezeigt, so befindet sich der Antrag in der Diskussionsphase. Hier können Verbesserungsvorschläge, alternative Anträge sowie quanifiziertes Feedback gegeben werden. Quantifiziertes Feedback bedeutet, dass man die Abstimmung in einem Antrag an Voraussetzungen wie gewünschte oder unerwünschte Änderungen knüpfen kann.

[12] Designated Verifier Proofs wurden in dieser Vorlesung nicht behandelt, der interessierte Leser möge im Internet nach dem Begriff suchen. Es handelt sich hierbei um Beweise, die nur eine bestimmte Person überzeugen, aber niemanden sonst, d.h. die Person kann den Beweis auch nicht weitergeben. Gerade im e-voting ist dies eine brauchbare Primitive, mit der man jemanden von einer Aussage überzeugen kann, ohne ihn durch das Wissen, dass die Aussage wahr ist, erpressbar zu machen.

Eingefrohren In dieser Phase wird anhand des Feedbacks in der Diskussionsphase entschieden, ob über einen Antrag abgestimmt wird oder nicht.

Abstimmung In dieser Phase findet nun die eigentliche Abstimmung statt, über eine Art Präferenzwahlverfahren nach der in dieser Vorlesung nicht behandelten Schulze-Methode [Sch11]. Die Stimmabgabe erfolgt dabei offen oder über ein Pseudonym nach Anmeldung mit dem eigenen Account beim Wahlserver. Der Wähler kann sein Stimmgewicht zu einem bestimmten Thema dauerhaft delegieren. In Abstimmungen zu diesem Thema erhält der entsprechende Proxy dann automatisch das Stimmgewicht des Wählers. Dieses Stimmgewicht kann der Wähler dem Proxy aber auch jederzeit wieder entziehen.

7.3.2.2 Sicherheit des Verfahrens

Obwohl Liquid Feedback kein kryptographisches Wahlverfahren ist, soll es hier der Vollständigkeit halber einen kurzen Überblick über die vorhandenen Sicherheitsmechanismen und Eigenschaften geben. Eine Sicherheitsbewertung des Verfahrens liefert dieser Abschnitt allerdings nicht, wer genaueres wissen möchte, sei auf die Implementierung und auf die Internetseite des Projekts verwiesen.

Wahlgeheimnis, Quittungsfreiheit und Nicht-Erpressbarkeit Das Wahlgeheimnis gehört nicht zu den Hauptzielen des Wahlverfahrens. Liquid Feedback ist für offene Wahlen konzipiert in der Form, dass die Datenbank der Stimmabgabe nach der Wahl veröffentlicht wird. Allerdings besteht die Möglichkeit, Pseudonyme zu verwenden. Das Wahlgeheimnis ist dann so sicher wie das Pseudonym, sofern die Abgabe der Stimme über eine verschlüsselte Verbindung erfolgt. Quittungsfreiheit und Nicht-Erpressbarkeit sind somit nicht erfüllt, der Wähler kann sein Pseudonym vor dem Angreifer offenlegen und ihm beweisen, dass er die zugehörigen Zugangsdaten besitzt. Auch bei diesem Wahlverfahren lernt der Wähler-PC die Stimme.

Integrität und Verifizierbarkeit Nach der Wahl wird die Datenbank der Stimmabgabe offengelegt, um für Transparenz zu sorgen. Dadurch könnte theoretisch das Wahlergebnis sowie die eigene Stimmabgabe von jedem überprüft werden, sofern nachprüfbar wäre, dass alle Pseudonyme zu wahlberechtigten Personen gehören, der Wahlserver keine Stimmen für Pseudonyme sich enthaltenter Wähler abgibt und niemand mehrere Pseudonyme besitzt. Durch das Fehlen kryptographischer Maßnahmen, die die Stimmabgabe nachvollziehbar und unabhängig von der Wahlsoftware mit dem Ergebnis in Verbindung bringen, haben wir keine Software-Independence. Durch die offene Stimmabgaben kann eine Manipulation zwar in den meisten Fällen von den Wählern erkannt, aber nicht nachgewiesen werden. Der Source Code der Implementierung ist veröffentlicht und kann von jedem interessierten überprüft werden. Das überprüfen von Software auf Sicherheitsmängel ist allerdings nicht-trivial, theoretisch sogar unmöglich (Man kann das bekannte Halteproblem darauf reduzieren). Außerdem ist es schwer, öffentlich verifizierbar zu machen, dass auf dem Wahlserver tatsächlich die richtige Software läuft.

Die Integrität der Wahl sowie das Erkennen und Nachweisen einer Manipulation hängt damit im Wesentlichen von einer verantwortungsbewußten Administration der Wahlserver ab und muss auf Serverseite (sowohl bei der Verteilung von Zugangsdaten an die Wähler als auch bei der Wahl selbst) durch Maßnahmen wie Sicherheitsaudits, regelmäßige Überprüfungen durch Administratoren und Auditoren, Sicherheitssoftware auf dem Wahlserver u.s.w. unterstützt werden.

Als Fazit lässt sich sagen, dass Liquid Feedback für hochsicherheitskritische Wahlen wie Parlamentswahlen in seiner aktuellen Form nicht empfehlenswert und vermutlich, schon allein wegen der Abwesenheit des Wahlgeheimnisses, auch nicht gedacht ist. Für Wahlen mit geriner Motivation zu Stimmkauf oder Manipulation, z.B. in Vereinen oder Organisationen, die sich intern in einer Liquid Democracy organisieren wollen, kann es aber genau wie das in Abschnitt 7.3.3 beschriebene Verfahren durchaus eine Option darstellen.

7.3.3 Agora

TODO

Literaturverzeichnis

[Adi] Adida, Ben: *Helios: Web-based Open-Audit Voting.*

[BHK+09] Bohli, Jens-Matthias, Christian Henrich, Carmen Kempka, Jörn Müller-Quade und Stefan Röhrich: *Enhancing electronic voting machines on the example of Bingo voting.* IEEE Transactions on Information Forensics and Security, 4(4):745–750, 2009.

[BHMQ+08] Bär, Michael, Christian Henrich, Jörn Müller-Quade, Stefan Röhrich und Carmen Stüber: *Real World Experiences with Bingo Voting and a Comparison of Usability.* Workshop On Trustworthy Elections, WOTE 2008, 2008.

[BMN+] Benaloh, Josh, Tal Moran, Lee Naish, Kim Ramchen und Vanessa Teague: *Shuffle-Sum: Coercion-Resistant Verifiable Tallying for STV Voting.*

[BMQR07] Bohli, Jens-Matthias, Jörn Müller-Quade und Stefan Röhrich: *Bingo Voting: Secure and Coercion-Free Voting Using a Trusted Random Number Generator.* In: Alkassar, Ammar und Melanie Volkamer (Herausgeber): *VOTE-ID*, Band 4896 der Reihe *Lecture Notes in Computer Science*, Seiten 111–124. Springer, 2007.

[BSI08] BSI: *Basissatz von Sicherheitsanforderungen an Online-Wahlprodukte.* 2008.

[bwg13] *Bundeswahlgesetz.* 2013.

[bwo12] *Bundeswahlordnung.* 2012.

[CCC+10] Carback, Richard, David Chaum, Jeremy Clark, John Conway, Aleksander Essex, Paul S. Herrnson, Travis Mayberry, Stefan Popoveniuc, Ronald L. Rivest, Emily Shen, Alan T. Sherman und Poorvi L. Vora: *Scantegrity II municipal election at Takoma Park: the first E2E binding governmental election with ballot privacy.* In: *Proceedings of the 19th USENIX conference on Security*, USENIX Security'10, Seiten 19–19, Berkeley, CA, USA, 2010. USENIX Association.

[CGH04] Canetti, Ran, Oded Goldreich und Shai Halevi: *The random oracle methodology, revisited.* J. ACM, 51(4):557–594, Juli 2004.

[CGS97] Cramer, Ronald, Rosario Gennaro und Berry Schoenmakers: *A Secure and Optimally Efficient Multi-Authority Election Scheme.* Seiten 103–118. Springer-Verlag, 1997.

[CP93] Chaum, David und Torben P. Pedersen: *Wallet Databases with Observers.* In: *Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '92, Seiten 89–105, London, UK, UK, 1993. Springer-Verlag.

[Dam99] Damgård, Ivan: *Commitment schemes and zero-knowledge protocols.* In: *Lectures on Data Security*, Seiten 63–86. Springer, 1999.

[ElG85] ElGamal, Taher: *A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms.* In: George Robert Blakley, David Chaum (Herausgeber): *Advances in Cryptology Proceedings of CRYPTO 84*, Seiten 10–18. Springer-Verlag, Berlin, Heidelberg, 1985.

[For02] Ford, Bryan: *Delegative Democracy*, 2002.

[FS86] Fiat, Amos und Adi Shamir: *How to Prove Yourself: Practical Solutions to Identification and Signature Problems.* In: Odlyzko, Andrew M. (Herausgeber): *CRYPTO*, Band 263 der Reihe *Lecture Notes in Computer Science*, Seiten 186–194. Springer, 1986.

[GA10] Green-Armytage, James: *Voluntary Delegation as the Basis for a Future Political System*, 2010.

[GBR] Geiselmann, Willi, Jens-Matthias Bohli und Stefan Röhrich: *Public Key Kryptographie.* Skript zur Vorlesung Public Key Kryptographie, verfügbar auf http://www.iks.kit.edu/index.php?id=asv-ws13.

[Gjø10] Gjøsteen, Kristian: *Analysis of an internet voting protocol.* IACR Cryptology ePrint Archive, 2010:380, 2010.

[Hen12] Henrich, Christian: *Improving and Anaysing Bingo Voting.* Dissertation am Karlsruher Institut für Technologie, 2012.

[JCJ05] Juels, Ari, Dario Catalano und Markus Jakobsson: *Coercion-resistant electronic elections.* In: *WPES '05: Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, Seiten 61–70, New York, NY, USA, 2005. ACM.

[Kem12] Kempka, Carmen: *Coercion-resistant electronic elections with write-in candidates.* In: *Proceedings of the 2012 international conference on Electronic Voting Technology/Workshop on Trustworthy Elections*, EVT/WOTE'12. USENIX Association, 2012.

[KTV12] Küsters, R., T. Truderung und A. Vogt: *A Game-Based Definition of Coercion-Resistance and its Applications.* Journal of Computer Security (special issue of selected CSF 2010 papers), 20(6/2012):709–764, 2012.

[liq] *Liquid Feedback.* http://liquidfeedback.org/.

[MM97] Mayers, Dominic und Université De Montréal: *Unconditionally Secure Quantum Bit Commitment is Impossible, Phys.* Rev. Lett, 1997.

[MN06] Moran, Tal und Moni Naor: *Receipt-Free Universally-Verifiable Voting with Everlasting Privacy.* In: *In CRYPTO*, Seiten 373–392. Springer-Verlag, 2006.

[Ped91] Pedersen, Torben Pryds: *A threshold cryptosystem without a trusted party.* In: *Proceedings of the 10th annual international conference on Theory and application of cryptographic techniques*, EUROCRYPT'91, Seiten 522–526, Berlin, Heidelberg, 1991. Springer-Verlag.

[PH10] Popoveniuc, Stefan und Benjamin Hosp: *An Introduction to PunchScan.* In: Chaum, David, Markus Jakobsson, Ronald L. Rivest, Peter Y. A. Ryan, Josh Benaloh, Miroslaw Kutylowski und Ben Adida (Herausgeber): *Towards Trustworthy Elections*, Band 6000 der Reihe *Lecture Notes in Computer Science*, Seiten 242–259. Springer, 2010.

[PS07] Popoveniuc, Stefan und Jonathan Stanton: *Undervote and pattern voting: Vulnerability and a mitigation technique.* In: *Preproceedings of the 2007 IAVoSS Workshop on Trustworthy Elections (WOTE 2007*, 2007.

[Riv06] Rivest, Ronald L: *The ThreeBallot voting system.* Unpublished draft, 2006.

[Röh07] Röhrich, Stefan: *Bingo Voting – Technische Kurzbeschreibung.* Europäisches Institut für Systemsicherheit, Universität Karlsruhe, Karlsruhe, 2007.

[Sch11] Schulze, Markus: *A new monotonic, clone-independent, reversal symmetric, and condorcet-consistent single-winner election method.* Social Choice and Welfare, 36(2):267–303, 2011.

[Sho97] Shor, Peter W: *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer.* SIAM journal on computing, 26(5):1484–1509, 1997.