Zero-knowledge proof für n-te Potenzen (Paillier)

Seminar E-Voting
Herbstsemester 2009

Halm Reusser
Hochschule Rapperswil
Oberseestrasse 10
CH-8640 Rapperswil
hreusser@hsr.ch

Prof. Dr. Andreas Steffen*
Hochschule Rapperswil
ITA, Institut für Internet-Technologien und Anwendungen
Oberseestrasse 10
CH-8640 Rapperswil
andreas.steffen@hsr.ch

ABSTRACT

In dieser Seminararbeit zum Gesamtthema **E-Voting** wird ein **Zero-knowledge proof** für n-te Potenzen anhand des **Paillier**-Kryptosystems vorgestellt.
Als Erstes folgt ein kleiner Überblick, wo ein Zero-knowledge proof im Kontext von E-Voting angewendet wird. Danach wird das Konzept des Zero-knowledge proof auf einer abstrakt-konzeptuellen Ebene betrachtet, sowie an einem einfachen kryptografischen Beispiel erläutert. Des weiteren werden die Grundlagen des Paillier-Kryptosystems — Prinzip, Schlüsselgenerierung, Verschlüsselung, Entschlüsselung — zusammengefasst.
Der Hauptteil dieser Arbeit besteht aus einem konkreten Beispiel anhand der **Überprüfung der Stimmabgabe**, wobei durch das Zero-knowledge proof Protokoll — Commitment, Challenge, Response — aufgedeckt werden soll, ob eine Stimme gültig ist.

*Advisor

1. EINFÜHRUNG

Peggy kennt ein Geheimnis. Sie möchte Viktor davon überzeugen, dass sie im Besitz eines geheimen Codes ist. Peggy überlegt sich, Viktor den geheimen Code aufzuschreiben, so könnte sie ihm beweisen, dass sie das Geheimnis kennt. Sie ist sich aber bewusst, dass wenn Viktor im Besitz des geheimen Codes ist, diesen jeder weiteren Person geben kann, was Peggy unbedingt vermeiden will.

Peggy steht vor einem Dilemma. Das Konzept des Zero-knowledge proof könnte ihr helfen.

Ein Zero-knowledge proof ist ein Protokoll aus dem Bereich der Kryptografie. Bei einem Zero-knowledge proof kommunizieren zwei Parteien — der Beweiser und der Verifizierer — miteinander. Der Beweiser (*Prover*) überzeugt dabei den Verifizierer (*Verifier*) mit einer gewissen Wahrscheinlichkeit davon, dass er ein Geheimnis kennt, ohne dabei Informationen über das Geheimnis selbst bekannt zu geben.

Der interaktive Zero-knowledge proof hat üblicherweise die Charakteristik eines Challenge-Response-Protokolls, wobei der Beweiser und der Verifizierer verschieden Mitteilungen austauschen und der Verifizierer am Ende *akzeptiert* oder *zurückweist*.

Eine typische Runde des Zero-knowledge proof besteht aus einem «Commitment» vom Beweiser an den Verifizierer, gefolgt von einer «Challenge» vom Verifizierer an den Beweiser und dessen «Response». Solche Runden können ein mehrfaches wiederholt werden. Aufgrund der Antwort des Beweisers akzeptiert oder verweigert der Verifizierer.

Alle interaktiven Beweise, mit denen man einzig und allein die Richtigkeit einer Aussage beweisen kann, nennt man Zero-knowledge proof.

1.1 Zero-knowledge proof im Kontext von E-Voting

Zero-knowledge proofs kommen generell überall in einem E-Voting System zur Anwendung, wo eine Information verschlüsselt übergeben wird und überprüft werden muss, ob die übergebene, verschlüsselte Information *wellformed* ist, d.h., eine gültige Information ist, ohne die Information selbst offenzulegen.

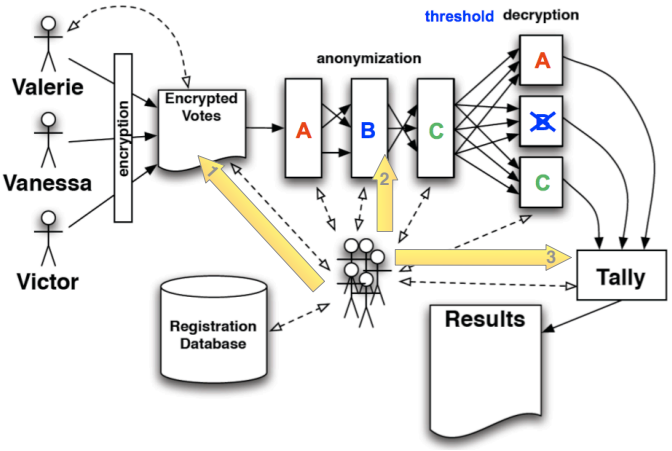

Abbildung 1: E-Voting System [17]

Die Grafik 1 zeigt Teilprozesse im Umfeld von E-Voting, wo es erforderlich ist, die Verarbeitung überprüfen zu können:

1. «Validity proof of the ballot». Jeder Wähler wie auch unabhängige Stelle kann (mit hoher Wahrscheinlichkeit) überprüfen, dass bei der Stimmabgabe nicht betrogen wurde. Dieser Anwendungsfall wird anhand eines Beispiels im Abschnitt 4 veranschaulicht.

2. «Mixnet Verification». Jeder Wähler wie auch jede unabhängige Stelle kann (mit hoher Wahrscheinlichkeit) überprüfen, ob die Anonymisierung korrekt ausgeführt wurde.

3. «Partial Decryption Verification». Jeder Wähler wie auch jede unabhängige Stelle kann (mit hoher Wahrscheinlichkeit) überprüfen, ob die Teilentschlüsselungen des Endresultates (*Tally*) durch die Parteien richtig vorgenommen wurde. D.h., die Parteien beweisen, dass der dabei verwendete Teilschlüssel richtig angewendet wurde und damit auch die Teilentschlüsselung korrekt ist.

Cramer et al. [9] sowie Adida et al. [17] haben Papers zum Thema *electronic Voting* veröffentlicht.

1.2 Abstraktes Zero-knowledge proof Beispiel

Schauen wir uns ein intuitives Beispiel mit dem Namen *Alibabas Höhle* [16] an, siehe Abbildung 2.

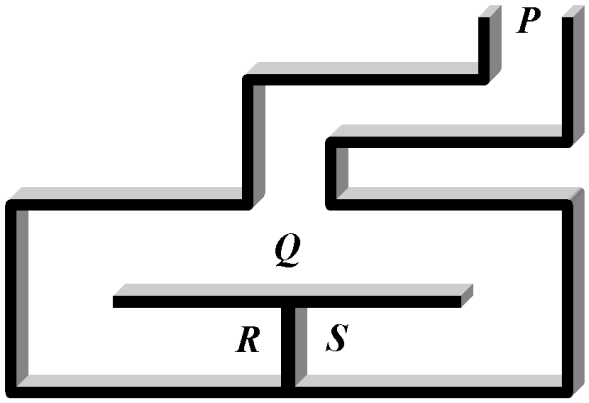

Abbildung 2: Alibabas Höhle [18]

Peggy will Victor davon überzeugen, dass sie den geheimen Spruch «Sesam öffne dich» kennt, um das Tor bei $R|S$ zu öffnen, ohne dabei Victor den geheimen Spruch zu verraten. Eine typische Runde läuft nun folgendermassen ab:

1. Victor bleibt bei Punkt P stehen. Peggys «Commitment» besteht darin, entweder zu R oder S zu gehen.

2. Nun geht Victor zum Punkt Q. Seine «Challenge» an Peggy zeichnet sich aus, indem er ihr zuruft, ob sie links oder rechts aus dem Ausgang kommen soll.

3. Peggys «Response» liegt vor, indem sie — falls notwendig mithilfe des Spruches — aus dem richtigen Ausgang kommt.

Falls Peggy den geheimen Spruch nicht kennt, besteht nur eine 50%-Chance, dass sie den richtigen Ausgang nimmt. Victor kann nun das Szenario in mehreren Runden wiederholen, bis er überzeugt ist, dass Peggy den geheimen Spruch kennt. Egal wie oft Victor den Beweis wiederholt, er bekommt keine Information, welche auf den geheimen Spruch deuten könnte.

Die Gewissheit, dass Peggy das Geheimnis kennt, steigt mit jeder Runde, in der Peggy zum richtigen Ausgang herauskommt. Nach einer Runde beträgt die Sicherheit 50%. Für jede weitere Runde n gilt:

$$\left(1 - 2^{-n}\right) \cdot 100\%$$

Nach beispielsweise 10 Runden hat Victor eine Sicherheit von 99.9%, dass Peggy den geheimen Spruch kennt.

2. ZERO-KNOWLEDGE PROOF

In diesem Abschnitt wird der Zero-knowledge proof [7] und seine Eigenschaften beschrieben.

Ein Zero-knowledge proof muss folgende drei Charakteristiken [7] erfüllen:

Vollständigkeit (*Completeness*) Der Verifizierer akzeptiert den Beweis immer, falls die Beweisführung richtig war und der Beweiser sowie der Verifizierer das Protokoll befolgt haben.

Zuverlässigkeit (*Soundness*) Der Verifizierer lehnt den Beweis immer ab, falls die Beweisführung falsch war und der Verifizierer das Protokoll befolgt.

Keine Erkenntnis (*Zero-knowledge*) Der Verifizierer lernt nichts vom Beweiser — ausgenommen, ob die Beweisführung richtig ist — was er nicht auch ohne den Beweiser wusste, sogar wenn der Verifizierer das Protokoll nicht befolgt (solange es der Beweiser tut). In einem Zero-knowledge proof ist es nicht möglich, dass der Verifizierer einem Dritten die gleiche Beweisführung vorlegt.

Die folgende formelle Definition eines Zero-knowledge proofs stammt aus Wikipedia [7].

Zero-knowledge proofs sind keine Beweise im Sinne der Mathematik, weil eine kleine Wahrscheinlichkeit, der *soundness error* besteht, dass der Beweiser den Verifizierer von einer falschen Tatsache überzeugen kann. In anderen Worten: Zero-knowledge proofs sind eher probabilistisch, denn deterministisch.

Eine formelle Definition des Zero-knowledge proof benutzt ein Computermodell, das wohl bekannteste ist das Modell der Turingmaschinen. Seien P, V und S Turingmaschinen. Ein interaktives proof System mit (P, V) für eine Sprache L ist zero-knowledge, falls für jeden probabilistischen polynomialzeit (PPT) Verifizierer $\hat{V}$, ein erwarteter PPT-Simulator S existiert, sodass:

$$\forall x \in L, z \in \{0,1\}^*, View_{\hat{V}}[P(x) \leftrightarrow \hat{V}(x,z)] = S(x,z)$$

Der Beweiser P wird betrachtet, als hätte er unendlich viel Rechenpower, beispielsweise eine probabilistische Turingmaschine. Intuitiv sagt die Definition aus, dass ein interaktives proof System (P, V) zero-knowledge ist, falls für jeden Verifizierer $\hat{V}$ ein effizienter Simulator S existiert, der die Konversation zwischen P und V für jeden Input reproduzieren kann. Der zusätzliche String z in der Definition spielt die Rolle des «Vorwissens». Die Definition impliziert, dass $\hat{V}$ keinen «Vorwissen»-String z benutzen kann, um durch die Konversation zwischen ihm und P an irgendwelche Information zu gelangen, weil wir verlangen, dass wenn S den «Vorwissen»-String z auch besitzt, die Konversation zwischen $\hat{V}$ und P reproduzieren kann.

Diese Definition ist für *perfect zero-knowledge*. *Computational zero-knowledge* wird erreicht, indem gefordert wird, dass die Sichten von $\hat{V}$ und S nur *computational* nicht zu unterscheiden sind, bei gegebenem zusätzlichen String z.

Oded Goldreich hat ein sehr umfassendes Paper [11] — Zero-knowledge proofs: 20 Jahre nach seiner Erfindung — verfasst. In dieser Arbeit werden auch verschiedene Varianten des Zero-knowledge proofs diskutiert, u.a. *Parallel Zero-knowledge proof*, wobei die Interaktion parallelisiert wird, was in der Praxis als sicher angesehen wird, obwohl nicht bewiesen ist, dass es tatsächlich zero-knowledge ist. Oded Goldreich vergleicht auch Varianten wie *Honest verifier vs. general cheating verifier*, *Statistical vs. computional zero-knowledge*, *Non-Interactive zero-knowledge* [4] etc.

2.1 Kryptografisches Beispiel

Das von Fiat-Shamir vorgestellte Verfahren [3] besitzt eine zero-knowledge Eigenschaft. Der Algorithmus wird beispielsweise auf Dekoderkarten von Pay-TV eingesetzt. Die Idee beruht darauf, dass es praktisch unmöglich ist, Quadratwurzeln in $\mathbb{Z}_n$ effizient zu berechnen. Es ist genau so schwer wie die Primfaktorzerlegung von n. Wir betrachten ein vereinfachtes Fiat-Shamir Protokoll [12].

Für die Initialisierung des Verfahrens werden zuerst die Schlüssel erzeugt:

1. Um Victor — den Verifizierer — von ihrem Geheimnis zu überzeugen, wählt Peggy — die Beweiserin — zwei grosse, verschiedene Primzahlen p und q und veröffentlicht das Produkt $n = p \cdot q$.
2. Peggy denkt sich eine geheime Zahl $s \in \mathbb{N}$ und gibt $v = s^2 \bmod n$ an Victor bekannt.

In der Anwendungsphase möchte nun Peggy, Victor davon überzeugen, dass sie im Besitz der geheimen Zahl s ist, ohne s preiszugeben. Dazu schauen wir uns den Ablauf in Abbildung 3 an:

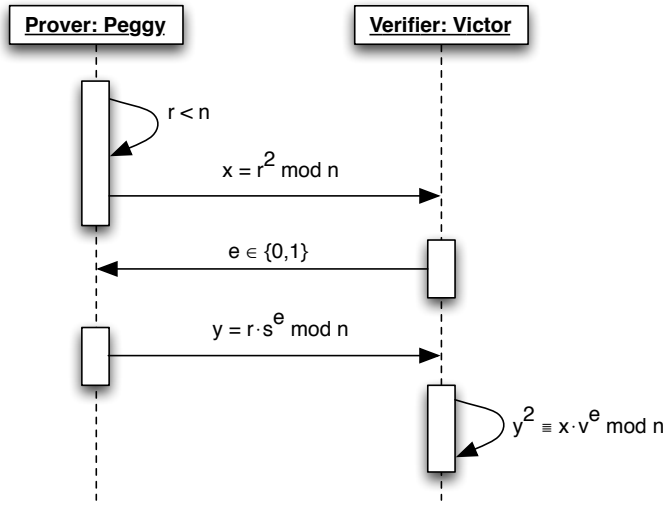

Abbildung 3: Vereinfachtes Fiat-Shamir Protokoll

Commitment Peggy wählt zufällig eine Zahl $r < n$ und sendet $x = r^2 \bmod n$ an Victor.

Challenge Victor wählt zufällig eine Zahl $e \in \{0, 1\}$ und sendet diese an Peggy.

Response Peggy berechnet und sendet $y = r \cdot s^e \bmod n$ an Victor.

Verifikation Victor akzeptiert den Beweis, falls $y^2 \equiv x \cdot v^e \bmod n$ gilt.

Die Gewissheit, dass Peggy das Geheimnis kennt, beträgt 50%. Eine Betrügerin kennt nie beide antworten, also für $e = 0$ und $e = 1$, wenn Victor $e = 0$ wählt, will er $y^2 \equiv x \bmod n$ sehen, ansonsten, bei $e = 1$, $y^2 \equiv x \cdot v \bmod n$. Falls Peggy beide Antworten geben kann, kennt sich auch $\sqrt{v}$, nämlich die geheime Zahl s:

$$\begin{aligned} y^2 &\equiv x \cdot v^e \bmod n \\ r^2 \cdot s^{e^2} &\equiv r^2 \cdot v^e \bmod n \\ s^2 &\equiv v \bmod n \\ s &\equiv \sqrt{v} \bmod n \end{aligned}$$

Weitere Beispiele, welche für einen Zero-knowledge proof verwendet werden können sind der Hamiltongraph oder der Graphisomorphismus [19, 15]. Generell kann jedes **NP** statement [20] verwendet werden.

3. PAILLIER-KRYPTOSYSTEM

Das Paillier-Kryptosystem [5] wurde von Pascal Pallier im Jahre 1999 erfunden [14].

Dieses Kryptosystem ist ein probabilistischer asymmetrischer Algorithmus für Public Key Kryptografie. Es basiert auf der Annahme, dass es schwer ist, *n-th residue* Klassen zu berechnen.

Um *n-th residue* zu verstehen, sehen wir uns die Definition für *quadratic residue* an [6]:

Eine Ganzzahl q wird als *quadratic residue* modulo n bezeichnet, falls sie kongruent zu einem Quadrat (modulo n) ist, d.h., falls eine Ganzzahl x existiert und Folgendes gilt:

$$x^2 \equiv q \bmod n$$

Auf das Paillier-Kryptosystem angewendet heisst das, dass aus dem Wert $r^n \bmod n^2$ nicht auf r zurück geschlossen werden kann, da das Ziehen der n-ten Wurzel mod n^2 eine extrem aufwendige Funktion ist.

Eine Eigenschaft des Paillier-Kryptosystems, welche vor allem im E-Voting Kontext grosse Bedeutung hat, ist die Homorphie. Bei Paillier handelt sich um eine *additive* Homorphie. Dies bedeutet, falls man die verschlüsselten Ciphertexte von m_1 und m_2 hat auch den verschlüsselten Ciphertext der Summe $m_1 + m_2$ berechnen kann.

Eine Generalisierung des Paillier-Kryptosystems für die Anwendung im E-Voting ist im Paper von Ivan Damgard et al. [10, 13] beschrieben.

3.1 Schlüsselgenerierung

Die Schlüsselgenerierung beim Paillier-Kryptosystem läuft in folgenden Schritten ab:

1. Seien $p, q \in \mathbb{N}$ Primzahlen und $n = p \cdot q$, wobei p und q voneinander unabhängig sein müssen, so das gilt $gcd(n, \varphi(n)) = 1$, wobei $\varphi(n)$ die eulersche φ-Funktion ist [2]. Dies kann sichergestellt werden, indem p und q von gleicher Länge sind.
2. Bestimme die Carmichael Funktion [1] $\lambda(n) = lcm(p - 1, q - 1)$.
3. Wähle eine zufällige Zahl $g \in \mathbb{Z}^*_{n^2}$, was bedeutet, dass g im Modulo n^2 System $GF(n^2)$ liegt und koprime zu n^2 ist.
4. Berechne $\mu = \left(\frac{g^\lambda \bmod n^2 - 1}{n}\right)^{-1} \bmod n$
5. Der öffentliche Schlüssel für die Verschlüsselung ist (n, g). Der private Schlüssel für die Entschlüsselung ist (λ, μ).

Falls man p und q von gleicher Länge wählt, können $g = n+1$, $\lambda = \varphi(n)$ und $\mu = \varphi(n)^{-1} \bmod n$ gewählt werden.

3.2 Verschlüsselung

Das Verschlüsseln einer Klartext Nachricht $m \in \mathbb{Z}_n$ funktioniert folgendermassen:

1. Wähle eine zufällige Zahl $r \in \mathbb{Z}^*_n$.
2. Berechne den Ciphertext $c = g^m \cdot r^n \bmod n^2$.

3.3 Entschlüsselung

Das Entschlüsseln einer Ciphertext Nachricht $c \in \mathbb{Z}_n$ zurück in den Klartext m, funktioniert folgendermassen:

$$m = \left(\frac{c^\lambda \bmod n^2 - 1}{n}\right) \cdot \mu \bmod n$$

4. ÜBERPRÜFUNG DER STIMMABGABE

Dieser Abschnitt beschreibt die Anwendung des Zero-knowledge proofs, um Stimmabgaben auf ihre Korrektheit zu überprüfen. Wir werden uns ein Beispiel anhand des Paillier Websimulators [21] schritt für schritt anschauen.

Das Szenario ist relativ einfach, es stehen eine Menge von Kandidaten zur Verfügung, jeder Wähler hat eine Stimme, die er einem oder keinem Kandidaten geben kann. Somit erfüllt eine gültige Stimme folgende zwei Kriterien:

- Die Stimme ist genau für einen Kandidaten oder keinen.
- Die Stimme zählt den Wert 1, d.h., es wurde nicht >1-mal für einen Kandidaten gestimmt.

Dies ist die Aufgabe des Zero-knowledge proofs, in welchem das Votingsystem — als Verifizierer — die Stimme des Wählers — als Beweiser — validiert. Der Wähler muss beweisen,

dass er eine korrekte Stimme abgegeben hat ohne dem Votingsystem offenzulegen, für wen er abgestimmt hat.

Der Zero-knowledge proof wird durch das Paillier-Kryptosystem implementiert, siehe Abschnitt 3.

4.1 Ausgangslage

Als Ausgangslage nehmen wir 4 Kandidaten $C_k, k \in \{0, 1, 2, 3\}$ an, wobei C_0 für keine Wahl steht. Eine Stimme m wird nun folgendermassen codiert:

Die Basis b ist 4, d.h., ein Kandidat kann total maximal $b - 1 = 3$ Stimmen erhalten.

	$C_3, m \cdot b^2$		$C_2,\ m \cdot b^1$		$C_1,\ m \cdot b^0$	
	m_3		m_2		m_1	
...	0	x	0	x	0	x

Tabelle 1: Codierung der Stimme

Falls für keinen Kandidaten C_0 gestimmt wird, ist die Stimme als $m_0 = m \cdot 0 = 0$ codiert. Für eine gültige Stimme wird nun genau ein $x = 1$ gesetzt, daraus ergeben sich folgende gültige Stimmen:

Stimme für Kandidat	Stimme	Binär	Dezimal
C_0	m_0	...000000	0
C_1	m_1	...000001	1
C_2	m_2	...000100	4
C_3	m_3	...010000	16

Tabelle 2: Gültige Stimmen

Für die eigentliche Abstimmung wird nun das Paillier-Kryptosystem initialisiert. In unserem Beispiel wählen wir die beiden Primfaktoren $p = 11$ und $q = 13$, woraus sich $n = 143$, sowie $n^2 = 20449$ ergibt. Als Generator wählen wir $g = 144$.

Für das Zero-knowledge proof Beispiel schauen wir uns einen ehrlichen Wähler V_1 sowie einen betrügerischen Wähler V_2 an.

V_1 stimmt einmal für C_1, V_2 stimmt zweimal für C_2:

Wähler	Kandidat i	Stimme m	Random r	chiff. Stimme c
V_1	1	1	5	5130
V_2	2	8	114	18412

Tabelle 3: Abstimmung

Wie die Stimme m verschlüsselt wird und die chiffrierte Stimme c ergibt, ist in Abschnitt 3.2 erläutert.

Abbildung 4 zeigt nun schematisch den Ablauf des Zero-knowledge proof Protokolls mit dem Zweck, zu überprüfen, ob die abgegebene Stimme gültig ist. Der Ablauf besteht aus fünf Abschnitten, «Vorbereitung», «Commitment», «Challenge», «Response» und zum Schluss noch die «Verifikation». Wir werden uns nun diese Abschnitte anhand unseres Beispiels genauer anschauen.

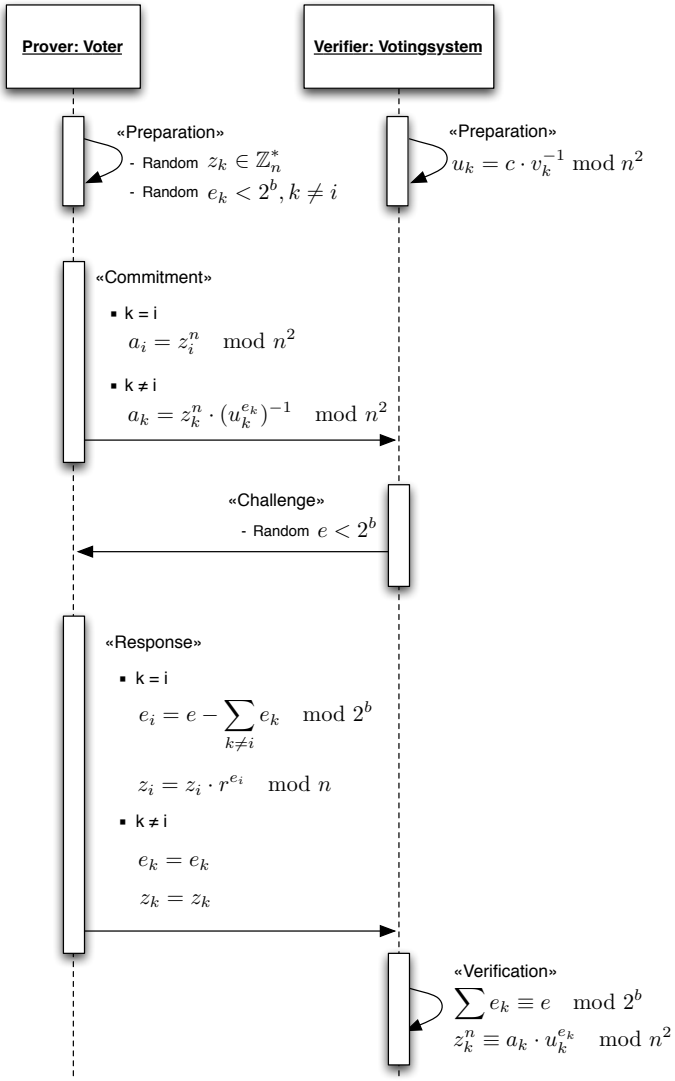

Abbildung 4: Überprüfung der Stimmabgabe

In der Abbildung 4 ist des Weiteren Folgendes definiert:

- $k \in \{0, \ldots, \#Canditates - 1\}$
- i: Kandidat, für welchen abgestimmt wurde
- $b = size_2(min(p, q)) - 1$

4.2 Vorbereitung

Als Erstes berechnet das Votingsystem alle gültigen Stimmzettelvarianten[1] $v_k = g^{m_k} \bmod n^2$, daraus ergibt sich folgende Tabelle:

v_0	v_1	v_2	v_3
1	144	573	2289

Tabelle 4: Gültige Stimmzettel

Andere Stimmzettel sind nicht erlaubt und sollen via den Zero-knowledge proof aufgedeckt werden.

Zur Erinnerung, die verschlüsselte Stimme c wird wie folgt berechnet:

$$c = g^m \cdot r^n \bmod n^2$$

Als Zweites stellt das Votingsystem alle

$$u_k = c \cdot v_k^{-1} \bmod n^2$$

für jede verschlüsselte Stimme auf dem Anschlagbrett (*Bulletinboard*) zur Verfügung:

	i	c	u_0	u_1	u_2	u_3
V_1	1	5130	5130	7704	15426	5416
V_2	2	18412	18412	2968	17983	16696

Tabelle 5: u_k

Jeder Wähler muss nun beweisen, dass alle u_k eine n-te Potenz darstellen. Dies kann der Wähler aber nur für $k = i$, da mit $m = m_i$ gilt:

$$c \cdot v_i^{-1} \bmod n^2 = r^n$$

In der Tabelle 5 sehen wir, dass dies für V_1 bei u_1 der Fall ist ($7704 \equiv 5^{143} \bmod 20449$), beim unehrlichen Wähler V_2 bei u_2 jedoch nicht ($17983 \not\equiv 114^{143} \bmod 20449$).

Damit das Votingsystem keine Rückschlüsse auf die Wahl ziehen kann, müssen alle u_k gleich behandelt werden und der Wähler darf seine geheime Zufallszahl r nicht verraten.

Jeder Wähler generiert zunächst $k = 4$ Zufallszahlen $z_k \in \mathbb{Z}_n^*$:

	z_0	z_1	z_2	z_3
V_1	118	45	15	108
V_2	84	125	24	23

Tabelle 6: z_k

[1] Alle Stimmzettelvarianten ohne die Maskierung durch ein zufälliges r^n

Des weiteren generiert jeder Wähler auch $k - 1 = 3$ Zufallszahlen $e_k < 2^b, k \neq i$, wobei hier $b = 3$ ist, und somit e_k im Bereich von $0 \ldots 7$ liegt:

	e_0	e_1	e_2	e_3
V_1	0	-	7	6
V_2	1	2	-	7

Tabelle 7: e_k

4.3 Commitment

Somit sind nun alle Vorbereitungen für das Zero-knowledge proof Protokoll abgeschlossen. Der erste Schritt besteht nun darin, dass der Beweiser dem Verifizierer ein «Commitment» macht.

Für das Teil-Commitment a_i, d.h., die Nachricht für den Kandidaten, den der Wähler tatsächlich gewählt hat, muss der Wähler sein zufälliges r maskieren, deshalb generiert er für diesen Fall:

$$a_i = z_i^n \bmod n^2$$

Man beachte, dass a_i eine echte n-te Potenz darstellt, was der Verifizierer jedoch nicht herausfinden kann.

Für alle anderen Fälle wird a_k rückwärts so konstruiert, damit es bei der Verifikation durchkommt, siehe Abschnitt 4.6:

$$a_k = z_k^n \cdot (u_k^{e_k})^{-1} \bmod n^2$$

Diese a_k's sind keine n-te Potenzen, was der der Verifizierer wiederum nicht herausfinden kann. Diese a_k's werden so generiert, damit sie bei der Überprüfung, siehe Abschnitt 4.6, durchkommen.

Das Resultat des «Commitments» für unser Beispiel ist in Tabelle 8 ersichtlich.

	i	a_0	a_1	a_2	a_3
V_1	1	11831	**16820**	19595	15772
V_2	2	12759	18089	**9314**	9023

Tabelle 8: Commitment a_k

4.4 Challenge

Der zweite Schritt des Zero-knowledge Protokolls besteht darin, indem der Verifizierer, dem Beweiser eine «Challenge» sendet, siehe Tabelle 9. Die Challenge e besteht aus einem zufälligen Wert $< 2^b$, wobei b die abgerundete Anzahl bits von p oder q ist, je nachdem welches kleiner ist.

In unserem Beispiel sind nur 8 Challengewerte möglich. In der Praxis wird man einen 2048-bit Modulus n wählen, wobei p und q jeweils 1024-bit gross sind. Damit sind dann 2^{1023} Challengewerte möglich.

	Challenge e
V_1	1
V_2	3

Tabelle 9: Challenge e

4.5 Response

Als Antwort zur «Challenge» sendet der Beweiser nun die «Response». Er muss alle e_k's offen senden, da er aber alle $e_k, k \neq i$ im «Commitment» schon benutzt hat, kann er diese nicht mehr verändern. Das Komplement e_i zu e, berechnet er folgendermassen:

$$e_i = e - \sum e_k \bmod 2^b, k \neq i$$

Danach sendet er alle e_k's, siehe Tabelle 10, dem Verifizierer.

	i	e_0	e_1	e_2	e_3
V_1	1	0	**4**	7	6
V_2	2	1	2	**1**	7

Tabelle 10: Response e_k

Des Weiteren sendet er alle z_k's, $k \neq i$ unverändert. Um den Verifizierer von der Gültigkeit seines Stimmzettels zu überzeugen, müsste der Beweiser für $z_i = r$ wählen — womit er aber das geheime r verraten würde. Deshalb haben wir beim «Commitment», siehe Abschnitt 4.3, das r maskiert. Um die Prüfung, siehe Abschnitt 4.6, zu bestehen, wird z_i auch maskiert:

$$z_i = z_i \cdot r^{e_i} \bmod n$$

Die gesendeten z_k's sind in Tabelle 11 ersichtlich.

	i	z_0	z_1	z_2	z_3
V_1	1	118	**97**	15	108
V_2	2	84	125	**19**	23

Tabelle 11: Response z_k

4.6 Verifikation

Als letzte Phase muss der Verifizierer die «Response» vom Beweiser überprüfen.

Zuerst wird verifiziert ob die Summe aller empfangenen e_k's mit der «Challenge» e übereinstimmt:

$$\sum e_k \equiv e \bmod 2^b$$

Tabelle 12 zeigt die Auswertung der ersten Verifikation. Kein Wähler kann hier als Betrüger entlarvt werden.

	$\sum e_k \bmod 2^b$	Challenge e
V_1	1	1
V_2	3	3

Tabelle 12: Verifikation Challenge e

Als Zweites wird überprüft, ob alle z_k's eine n-te Potenz sind, dafür wird folgende Gleichung überprüft:

$$z_k^n \equiv a_k \cdot u_k^{e_k} \bmod n^2$$

Die Auswertung ist in Tabelle 13 dargestellt. Es ist ersichtlich, dass der Wähler V_2 bei Kandidat $i = 2$ geschummelt hat, denn Kongruenz für z_i wird nur erzielt, falls $m = m_i$ ist.[2]

	...	$z_1^n \equiv a_1 \cdot u_1^{e_1} \bmod n^2$		$z_2^n \equiv a_2 \cdot u_2^{e_2} \bmod n^2$		...
V_1	...	15975	15975	8600	8600	...
V_2	...	15618	15618	8201	3090	...

Tabelle 13: Verifikation Commitment a_k **und Response** z_k

Der Beweis der Verifikationsgleichung, für $k = i$:

$$\begin{aligned}
z_i^n &\equiv a_i \cdot u_i^{e_i} \bmod n^2 \\
&\rightarrow z_i^n = z_i^n \cdot r^{e_i^n} \bmod n^2 \\
&\rightarrow a_i = z_i^n \bmod n^2 \\
&\rightarrow u_i = c \cdot (g^{m_i})^{-1} \bmod n^2 \\
z_i^n \cdot r^{e_i^n} &\equiv z_i^n \cdot \left(c \cdot (g^{m_i})^{-1}\right)^{e_i} \bmod n^2 \\
&\rightarrow c = g^m \cdot r^n \bmod n^2 \\
z_i^n \cdot r^{e_i^n} &\equiv z_i^n \cdot \left(g^m \cdot r^n \cdot (g^{m_i})^{-1}\right)^{e_i} \bmod n^2 \\
&\rightarrow m = m_i \\
z_i^n \cdot r^{e_i^n} &\equiv z_i^n \cdot r^{n^{e_i}} \bmod n^2
\end{aligned}$$

und $k \neq i$:

$$\begin{aligned}
z_k^n &\equiv a_k \cdot u_k^{e_k} \bmod n^2 \\
&\rightarrow a_k = z_k^n \cdot (u_k^{e_k})^{-1} \bmod n^2 \\
z_k^n &\equiv z_k^n \cdot (u_k^{e_k})^{-1} \cdot u_k^{e_k} \bmod n^2 \\
z_k^n &\equiv z_k^n \bmod n^2
\end{aligned}$$

5. PRAXIS

Das webbasierte Votingsystem Helios [8] bietet eine Implementation anhand des Paillier-Kryptosystems. Es sind einige Möglichkeiten implementiert, u.a. auch *Scratch & Vote*.

[2] Falls der unwahrscheinliche Fall $e_i = 0$ eintritt, kann ein Betrüger unerkannt bleiben Da er aber den gefälschten Stimmzettel abgeben muss, bevor er e_i ableiten kann, nimmt er ein grosses Risiko in Kauf erwischt zu werden.

6. REFERENCES

[1] Carmichael function from wikipedia.org, 10 2009. http://en.wikipedia.org/wiki/Carmichael_function.

[2] Euler's totient function from wikipedia.org, 10 2009. http://en.wikipedia.org/wiki/Euler's_totient_function.

[3] Fiat-Shamir Protokoll from de.wikipedia.org, 10 2009. http://de.wikipedia.org/wiki/Fiat-Shamir-Protokoll.

[4] Non-interactive zero-knowledge proof from wikipedia.org, 10 2009. http://en.wikipedia.org/wiki/Non-interactive_zero-knowledge_proof.

[5] Paillier cryptosystem from wikipedia.org, 10 2009. http://en.wikipedia.org/wiki/Paillier.

[6] Quadratic residue from wikipedia.org, 10 2009. http://en.wikipedia.org/wiki/Quadratic_residue.

[7] Zero-knowledge proof from wikipedia.org, 10 2009. http://en.wikipedia.org/wiki/Zero_knowledge_proof.

[8] B. Adida. Helios Voting System, 10 2009. http://www.heliosvoting.org/.

[9] R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. pages 103–118. Springer-Verlag, 1997.

[10] I. Damgard, M. Jurik, and J. B. Nielsen. A generalization of paillier's public-key system with applications to electronic voting. *P Y A Ryan*, page 03, 2003.

[11] O. Goldreich. Zero-knowledge twenty years after its invention, 2004.

[12] S. Kuz. Fiat shamir und zero knowdledge proofs. 2005.

[13] R. P. Of, I. Damgaard, R. Cramer, R. Cramer, I. Damgard, B. Schoenmakers, and B. Schoenmakers. Centrum voor wiskunde en informatica. In *Advances in Cryptology-CRYPTO'94, pp*, pages 174–187. Springer-Verlag, 1994.

[14] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. pages 223–238. Springer-Verlag, 1999.

[15] J. Pieprzyk, J. Seberry, and T. Hardjono. *Fundamentals of Computer Security*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[16] J.-J. Quisquater, M. Quisquater, M. Quisquater, M. Quisquater, L. C. Guillou, M. A. Guillou, G. Guillou, A. Guillou, G. Guillou, S. Guillou, and T. A. Berson. How to explain zero-knowledge protocols to your children. In *CRYPTO '89: Proceedings of the 9th Annual International Cryptology Conference on Advances in Cryptology*, pages 628–631, London, UK, 1990. Springer-Verlag.

[17] R. L. Rivest, A. C. Smith, and B. Adida. Advances in cryptographic voting systems. Technical report, 2006.

[18] RSA Conference. What are interactive proofs and zero-knowledge proofs?, 10 2009. https://365.rsaconference.com/docs/DOC-1374/.

[19] B. Schneier. *Applied cryptography (2nd ed.): protocols, algorithms, and source code in C*. John Wiley & Sons, Inc., New York, NY, USA, 1995.

[20] M. Sipser. *Introduction to the theory of computation: second edition*. PWS Pub., Boston, 2 edition, 2006. 96035322 Michael Sipser. Includes bibliographical references and index. Pt. 1. Automata and Languages. 1. Regular Languages. 2. Context-Free Languages – Pt. 2. Computability Theory. 3. The Church-Turing Thesis. 4. Decidability. 5. Reducibility. 6. Advanced Topics in Computability Theory – Pt. 3. Complexity Theory. 7. Time Complexity. 8. Space Complexity. 9. Intractability. 10. Advanced topics in complexity theory.

[21] A. Steffen. E-Voting Simulator based on the Paillier Cryptosystem, 10 2009. http://security.hsr.ch/msevote/paillier.