Leibniz Universität Hannover
Fakultät für Elektrotechnik und Informatik
Institut für Theoretische Informatik

Bachelorarbeit

Commitment-Verfahren

Commitment Schemes

Rico Seebonn
Matrikelnummer: 3216740

12. Oktober 2020

Erstprüfer:	Prof. Dr. Heribert Vollmer
Zweitprüfer:	PD Dr. Arne Meier
Betreuer:	PD Dr. Arne Meier

Inhaltsverzeichnis

1 Einleitung

In der heutigen Zeit verläuft ein großer Teil der Kommunikation zwischen Menschen digital ab. Nachrichten an Kontakte über Social Media, geschäftliche Entscheidungen zwischen Unternehmen via Videokonferenz, das Lesen und Verbreiten von Informationen und vieles mehr wird durch das Internet ermöglicht. Der Alltag der Menschen wird immer vernetzter, ob es Formulare sind, die nur noch online eingereicht werden können, oder auch Schulunterricht, welcher vorübergehend aufgrund einer Pandemie online stattfinden muss. Natürlich haben Internetnutzer dabei dieselbe Erwartung an ihre Privatsphäre, wie sie es im echten Leben auch beispielsweise an einem normalen Gespräch hätten: Der Inhalt der Unterhaltung mit dem Gesprächspartner bleibt privat. Andere Nutzer sollen nicht einfach die Nachrichten eines Gesprächs mitlesen dürfen. Daher gilt es die privaten Nachrichten und Verbindungen von Nutzern gegenüber Dritten unkenntlich zu machen.

Aus diesem Grund spielt die Verschlüsselung der Kommunikation im Internet eine wichtige Rolle. Die Kryptographie beschäftigt sich mit der Verschlüsselung von Informationen und dem dazugehörigen Themenbereich. Kryptographische Methoden erlauben die sichere Nutzung des Internets. Durch entsprechende Ver- und Entschlüsselungsmethoden erlaubt der Einsatz von Kryptosystemen gesendete Nachrichten privat zu halten. Kryptographische Signaturen an Nachrichten lassen den Empfänger wissen, dass die Nachricht wirklich vom Sender stammt und nicht manipuliert wurde. Für den Endnutzer passieren diese Anwendungen automatisch. Der Bereich und die Applikationen der Kryptographie wachsen stetig, so ermöglichen sie auch neuere Konzepte wie zum Beispiel die Kryptowährung Bitcoin. Commitment-Verfahren sind ein Teilgebiet der Kryptographie.

Münzwurfprotokoll

Die ersten Überlegungen zu einem Problem, mit der Notwendigkeit zur Nutzung eines Commitment-Verfahrens, hat Manuel Blum 1981 auf einer Konferenz gemacht [Blu81]. Er stellte das Problem vom Münzwurf via Telefon vor:

Das Ehepaar Alice und Bob will sich scheiden lassen. Weil sie vorher zusammen in einem Haus gelebt haben, werden sie sich bei manchen Möbelstücken nicht einig, wer sie behalten darf. Früher haben sie sich bei solchen Meinungsverschiedenheiten mit einem Münzwurf Abhilfe geschafft. Wenn Alice das Ergebnis des Münzwurfs von Bob richtig vorhersagen konnte durfte sie entscheiden, lag sie falsch durfte es Bob. Leider kann diese Lösung hier nicht angewendet werden, mittlerweile haben sie sich so verfeindet, dass sie den Anblick voneinander nicht ertragen können. Über das Telefon diese Spiel auszuführen ist für sie auch keine Option, weil sie sich nicht gegenseitig vertrauen. Wenn Alice vor dem Münzwurf ihre Voraussage macht, kann Bob einfach

immer sagen sie läge falsch, weil sie die Ausführung und das Ergebnis des Wurfs nicht sehen kann. Umgekehrt, sollte Bob zuerst das Ergebnis des Wurfs mitteilen, kann Alice immer behaupten natürlich die richtige Vorhersage gemacht zu haben.

Dieses Problem ist lösbar mit einem Commitment-Verfahren. Bei einem Commitment handelt es sich laut Duden um „das [Sich]bekennen, [Sich]verpflichten“ [Dud], und genau darum geht es in solchen Verfahren. Sie werden benutzt von Parteien, die sich zu einer Wahl oder einem Wert verpflichten und später diesen Wert mit anderen Parteien teilen wollen. Hierzu wird nur die Verpflichtung gesendet, welche den Wert geheim hält. Es ist den anderen Parteien also nicht möglich den Wert vorzeitig zu kennen und nur mit dem Einverständnis der sendenden Partei ist es ihnen später möglich den Wert aus der Verpflichtung zu ermitteln. Außerdem sollte es nicht möglich sein die Verpflichtung auf verschiedene Arten zu öffnen, sodass sich verschiedene Werte ergeben können. Andernfalls wäre es wortwörtlich keine Verpflichtung mehr.

Um den Münzwurf telefonisch durchzuführen, muss sich Alice zu Kopf oder Zahl verpflichten können. Sollte sich Alice zu ihrer Voraussage verpflichten und diese später aufdecken können, kann sich Bob sicher sein nicht betrogen zu werden, solange er die Ansage des Ergebnisses nach Erhalt von Alices Commitment macht. Die beiden überlegen deshalb ein Verfahren, das dies ermöglichen soll: Alice schreibt Kopf beziehungsweise Zahl auf ein Blatt Papier und schließt dieses in einen Tresor ein. Danach schickt sie den Tresor per Post an Bob. Sobald er den Tresor erhalten hat, beginnen sie ihr Telefonat. Bob führt den Münzwurf durch und teilt Alice das Ergebnis mit. Alice muss nun beweisen, ob sie sich zu Kopf oder Zahl festgelegt und damit gewonnen oder verloren hat. Dazu sagt Alice Bob die Zahlenkombination für den Tresor. Nun öffnet Bob den Tresor und überprüft was auf dem Zettel steht, um den Gewinner zu ermitteln.

In diesem analogen Beispiel scheint ein Commitment-Verfahren sehr aufwändig zu sein. Die digitale Nutzung solcher Verfahren erleichtert natürlich das Senden und Empfangen der Daten, wodurch die größte Herausforderung in dem Erstellen eines Commitments liegt. Denn Commitments sollen sowohl vor Empfänger, als auch Sender sicher sein. Fortan werden in dieser Arbeit zur besseren Lesbarkeit die bereits eingeführten Alice und Bob in ihren Rollen als Sender und Empfänger synonym benutzt.

Commitments besitzen zwei wichtige Sicherheitseigenschaften Hiding und Binding. Durch die Hiding-Eigenschaft ist es Bob ohne Freigabe von Alice nicht möglich den Wert hinter einem Commitment zu erfahren. In dem analogen Beispiel erfüllt der Tresor diese Eigenschaft symbolisch, Bob kann nicht in ihn hineinsehen und ihn nicht unbefugt öffnen. Binding bezieht sich darauf, dass Alice nicht den Wert hinter einer Verpflichtung unbemerkt ändern kann. So darf der Tresor beispielsweise nicht verschiedene Fächer besitzen, die sich bei verschiedenen Zahlenkombinationen öffnen, damit diese Eigenschaft gilt.

Diese Arbeit soll einen grundlegenden Überblick über das Thema Commitment-Verfahren verschaffen, in dem die häufig verwendet und bekannten Pedersen und ElGamal Commitment-Verfahren vorgestellt und untersucht werden und die Tauglichkeit von bekannten kryptographischen Methoden als Commitment-Verfahren betrachtet wird. Weil in digitalen Commitment-Verfahren die Sicherheit von Commitments auf mathematischen Problemen basieren, werden vorerst dazu mathematische sowie kryp-

tographische Grundlagen erarbeitet. Darauf aufbauend werden im folgenden Kapitel allgemeine Eigenschaften von Commtiment-Verfahren an einem generischen Beispiel behandelt. Anschließend wird untersucht, ob sie durch Verwendung des Kryptosystems RSA und auch kryptographischen Hashfunktionen realisiert werden können. Danach werden die bereits erwähnten Commitment-Verfahren Pedersen und ElGamal nacheinander definiert, Eigenschaften und Nutzung gezeigt und ein mögliches Szenario zur Verwendung gezeigt. Abschließend erfolgt eine Zusammenfassung der wichtigsten Erkenntnisse der Arbeit sowie ein Ausblick in weiterführende Themengebiete und speziellere Anwendungsfälle der Commitment-Verfahren.

2 Grundlagen

2.1 Funktionsweise

Digital werden Commitment-Verfahren in den drei Phasen Setup, Commit und Open umgesetzt in denen grundlegend jeweils Folgendes geschieht:

Setup: Es wird ein zufälliger öffentlicher Schlüssel erzeugt $CK \leftarrow \texttt{setup}(1^n)$, der Alice und Bob zur Verfügung steht. 1^n notiert eine Folge von n Einsen. CK wird für jedes Commitment neu generiert. Zusätzlich sollte in dem Fall, dass dieser Schlüssel falsch generiert beziehungsweise manipuliert worden sein könnte abgebrochen und neu gestartet werden.

Commit: Alice verpflichtet sich zu einer Nachricht m und berechnet das Commitment C mit $\texttt{commit}_{CK}(m) \rightarrow C$. Hierbei ist $\texttt{commit}_{CK}$ die von dem Commitment-Verfahren definierte Funktion ein Commitment in Abhängigkeit eines CKs zu erzeugen. Alice schickt C an Bob.

Open: Wenn Alice bereit ist Bob das Commitment C zu offenbaren, schickt sie m an ihn. Bob führt $\texttt{open}_{CK}(m) \rightarrow w$ aus, wobei $w \in \{0, 1\}$ ablehnen oder akzeptieren repräsentiert. Die Funktion `open` besteht in der Praxis oft darin zu überprüfen, ob $C = \texttt{commit}_{CK}(m)$ und dementsprechend anzunehmen.

Wie und ob die Sicherheitseigenschaften Hiding und Binding erfüllt sind, hängt von dem verwendeten Verfahren ab. Durch die Binding-Eigenschaft soll es Alice nicht möglich sein mehrere m zu finden, die dasselbe C erzeugen. Hiding bezieht sich darauf, dass es Bob nicht möglich ist m selbst zu finden und somit C zu brechen. Dazu wird grundsätzlich zur Berechnung eines Commitments C der Nachricht m eine Zufallszahl r beigefügt, der Anlass hierzu wird im nächsten Unterkapitel erklärt.

Das in Kapitel 3 eingeführte 1-Bit-Commitment-Verfahren illustriert das Tripel (Setup, Commit, Open) und die dazugehörigen Hiding- und Binding-Eigenschaften genauer. Dafür werden nun vorerst grundlegende statistische und mathematische Konzepte aus der Kryptographie thematisiert.

2.2 Kryptographie

Algorithmen

Zufallszahlen werden benötigt um ein Commitment ausreichend zu verbergen. Da Alice und Bob beide Zugriff auf das Verfahren der Verschlüsselung haben, wäre es für Bob

nicht schwer den Wert des Commitments ohne Zufallsinformationen zu erfahren. Er kann trivialerweise selbst ausprobieren jeden möglichen Wert zu verschlüsseln und diesen mit dem erhaltenen Commitment vergleichen. Abhängig von der Anzahl der möglichen Werte, die sich mit dem Commitment-Verfahren verschlüsseln lassen, kann Bob in kürzester Zeit zu einem Ergebnis kommen.

In der Realität sind Zufallszahlen oftmals Pseudozufallszahlen, das heißt sie werden von statistisch getesteten Algorithmen generiert, welche eine zufällig aussehende Verteilung erzeugen können (vlg. [Rot08, Kapitel 7.2]). In dieser Arbeit wird jedoch davon ausgegangen, dass *probabilistische Algorithmen* echte Zufallszahlen liefern.

Ein probabilistischer Algorithmus kann aufgrund einer Eingabe eine zufällig Ausgabe erzeugen. Solche Algorithmen können von probabilistischen Turingmaschinen ausgeführt werden. Knapp beschrieben sind Turingmaschinen eine Art primitiver Computer, die einen bestimmten Algorithmus repräsentieren. Abhängig von der Eingabe führen sie Rechenschritte auf einem Arbeitsband aus und liefern am Ende aller Rechenschritte eine Ausgabe zurück. Eine probabilistische Turingmaschine ist eine spezielle Turingmaschine mit der Fähigkeit zufällige Entscheidungen zu treffen. So besitzt sie nach jedem Rechenschritt für den darauffolgenden eine Auswahl zwischen zwei verschiedenen Rechenschritten von denen unabhängig von der Eingabe einer zufällig gewählt wird. Bildlich vorgestellt wirft diese Maschine jedes mal eine Münze und entscheidet so den nächsten Rechenschritt (vgl. [Rot08, Kapitel 6.2]).

Beispiel 1. *Eine probabilistische Turingmaschine T erhält die Eingabe „11111111“ und T hat in jedem Rechenschritt die Wahl die nächste 1 durch eine 0 zu ersetzen oder die 1 stehen zu lassen. Eine mögliche Ausgabe wäre dann „10000101“. Da T zufällige Entscheidung trifft, ist als Ausgabe jede achtstellige Folge aus Nullen und Einsen gleich wahrscheinlich.*

Genauere Einblicke in die Funktionsweisen und insbesondere formale Definitionen von Turingmaschinen können im Lehrbuch von Rothe [Rot08] nachgelesen werden.

Weil Zufallszahlen oft benötigt werden, sollen die erzeugenden probabilistischen Algorithmen effizient sein. Algorithmen mit polynomieller Laufzeit, auch Polynomialzeit-Algorithmen genannt, gelten als effizient. Ein Algorithmus hat polynomielle Laufzeit, wenn er bei Eingabe x höchstens eine Laufzeit von $O(|x|^n)$ mit $k \in \mathbb{N}$ hat [Buc04].

Vernachlässigbar

Im Abschnitt Algorithmen wurde beschrieben, dass Bob ein Commitment ohne Zufallsinformationen in Form einer Zufallszahl brechen kann, indem er probiert jeden Wert selber zu verschlüsseln und vergleicht. Doch selbst bei einem Commitment mit Zufallsinformationen ist Bob theoretisch in der Lage den Wert und die Zufallszahl zu erraten und so das Commitment zu brechen. Die Wahrscheinlichkeit mit der es Bob gelingt richtig zu raten, und deshalb die Hiding-Eigenschaft nicht mehr gilt, soll vernachlässigbar sein. Vernachlässigbare Funktionen beziehen sich hier auf diese Wahrscheinlichkeiten.

Definition 1 ([Gol01, Definiton 1.3.5])**.** *Eine Funktion $f\colon \mathbb{N} \to \mathbb{R}$ heißt* vernachlässigbar*, wenn für jedes Polynom $p\colon \mathbb{N} \to \mathbb{N}$, es eine natürliche Zahlt n_0 gibt, sodass für alle $n > n_0$ gilt*

$$f(n) < \frac{1}{p(n)}$$

Im Folgenden wird die Notation $negl(n)$, vom Englischen „negligible“, für eine vernachlässigbare Funktion verwendet.

Ununterscheidbarkeit

Besonders bezüglich der Hiding-Eigenschaft spielt Ununterscheidbarkeit bei Commitment-Verfahren eine wichtige Rolle. Bei Commitments zu verschiedenen Werten ist es für Bob nicht erkennbar, welche Werte hätten benutzt werden können. Aus diesem Grund bestehen Commitments aus dem eigentlichen Wert und zusätzlichen Zufallszahlen. Diese Zufallszahlen sollen garantieren, dass Bob kein Muster zwischen dem Commitment und dem Wert erkennen kann.

Ununterscheidbarkeit kann besser beschrieben werden mit den probabilistische Algorithmen U und V, die mit Eingabe x eine zufällige Ausgabe erzeugen. Die Wahrscheinlichkeit, dass U ein y ausgibt bei Eingabe x wird als $U_x(y)$ notiert. U_x ist die Wahrscheinlichkeitsverteilung aller Ausgaben U mit Eingabe x. Es werden nun beide Algorithmen U und V mit Eingabe x benutzt und zufällig ein y der beiden Ausgaben gewählt. Ein außenstehender Beobachter erhält x und y und bekommt die Aufgabe herauszufinden, ob U oder V benutzt wurde um y zu generieren. U und V sind ununterscheidbar, wenn der Beobachter das richtige Ergebnis nicht mit einer größeren Chance als 50% erraten kann.

Definition 2 ([DN08, Definition 2])**.** *Gegeben seien zwei Wahrscheinlichkeitsverteilungen P, Q. Die* statistische Distanz *dieser Verteilungen wird definiert als $SD(P,Q) = \sum_y \big|P(y) - Q(y)\big|$, wobei $P(y)$ und $Q(y)$ die Wahrscheinlichkeiten sind, dass P respektive Q den Wert y annimmt.*

Die statistische Distanz quantifiziert die Ähnlichkeit zweier Wahrscheinlichkeitsverteilungen folgendermaßen:

Beispiel 2. *Gegeben seien die Wahrscheinlichkeitsverteilungen A und B zweier Würfel, welche eine unterschiedliche Anzahl an Seiten besitzen. A ist die Wahrscheinlichkeitsverteilung eines üblichen sechsseitigen Würfels, somit ist die Chance bei einem Wurf ein $n \in [1,6]$ zu erzielen $\frac{1}{6}$ für jedes n. B entspricht der eines zehnseitigen Würfels, analog ist $\frac{1}{10}$ hier die Wahrscheinlichkeit ein $n \in [1,10]$ zu würfeln.*

Die statistische Distanz von A und B ist dann gegeben mit:

$$\begin{aligned} SD(A,B) &= \sum_n \big|A(n) - B(n)\big| \\ &= \big|A(1) - B(1)\big| + \ldots + \big|A(10) - B(10)\big| \\ &= 6 \cdot \left|\frac{1}{6} - \frac{1}{10}\right| + 4 \cdot \left|0 - \frac{1}{10}\right| \\ &= \frac{6}{15} + \frac{4}{10} \\ SD(A,B) &= \frac{4}{5} \end{aligned}$$

Definition 3 ([DN08, Definition 3])**.** *Gegeben seien zwei Wahrscheinlichkeitsverteilungen U, V.*

- *U, V sind* perfekt ununterscheidbar*, geschrieben $U \sim^p V$, wenn gilt $\forall x\ U_x = V_x$*
- *U, V sind* statistisch ununterscheidbar*, geschrieben $U \sim^s V$, wenn gilt $SD(U_x, V_x) \leq negl(x)$.*
- *U, V sind* algorithmisch ununterscheidbar*, geschrieben $U \sim^c V$, wenn für jeden probabilistischen Polynomialzeit Algorithmus D gilt: Sei $P_{U,D}(x)$ die Wahrscheinlichkeit, dass D die Ausgabe „A“ erzeugt mit Eingabe von U. Analog sei $P_{V,D}(x)$ die Wahrscheinlichkeit, dass D auch die Ausgabe „A“ erzeugt mit Eingabe von V. Dann ist $|P_{U,D}(x) - P_{V,D}(x)| \leq negl(x)$.*

Anschaulich beschrieben hat ein Außenstehender bei der Zuordnung von Ausgaben zweier Wahrscheinlichkeitsverteilungen U, V bei denen $U \sim^p V$ gilt keine andere Wahl als zu raten. Sollten U und V statistisch ununterscheidbar sein, kann er selbst mit unbegrenzter Rechenkraft nur mit einer vernachlässigbaren Wahrscheinlichkeit besser als zu 50% raten. U und V könnten stark verschieden sein und es könnte dennoch $U \sim^c V$ gelten, falls es für Außenstehende keine Möglichkeit gibt diese Unterschiede in Polynomialzeit zu erkennen.

Public-Key-Kryptosystem

Kryptosysteme sind bestimmte Verschlüsselungsverfahren, die benutzt werden um Daten sicher zwischen Kommunikationspartnern über ein unsicheres Medium zu übertragen. Angreifer, die das Übertragungsmedium abhören, erhalten nur verschlüsselte Daten aus denen sie nichts über den eigentlichen Inhalt auslesen können. Verschlüsselungsverfahren verschlüsseln einen Klartext m mit einem Schlüssel k zu einem Ciphertext c. Dieser Ciphertext lässt sich mit einem Schlüssel d wieder zu dem Klartext m entschlüsseln. Man redet von einem symmetrischen Verschlüsselungsverfahren, falls $k = d$ gilt. Wenn dies nicht gilt, handelt es sich um ein asymmetrisches Verschlüsselungsverfahren, auch Public-Key-Verschlüsselungsverfahren genannt mit k als öffentlichen und d als privaten Schlüssel. In Kapitel 4 wird anhand des Public-Key-Kryptoverfahren RSA gezeigt, dass

einige Public-Key-Kryptosysteme als Commiment-Verfahren benutzt werden können. Public-Key-Kryptosysteme sind folgendermaßen definiert:

Definition 4 ([KL07, Definition 10.1])**.** *Ein Public-Key-Verschlüsselungsverfahren ist ein Tupel aus den probabilistischen Polynomialzeit Algorithmen (S,E,D) mit den folgenden Eigenschaften:*

1. Schlüsselerzeugung S: *Erhält als Sicherheitsparameter 1^ℓ und produziert das Schlüsselpaar k und d. Geschrieben $S(1^\ell) = (k, d)$.*
2. Verschlüsselung E: *Erzeugt abhängig von einem Klartext m und dem öffentlichen Schlüssel k einen Ciphertext c. Geschrieben $E_k(m) = c$.*
3. Entschlüsselung D: *Entschlüsselt einen Ciphertext c anhand des privaten Schlüssels d zu einem Klartext m oder signalisiert das Scheitern der Entschlüsselung. Geschrieben $D_d(c) = m$.*

Für jedes l und damit jedes Schlüsselpaar (k, d) erzeugt von S und für jeden Klartext m gilt

$$S_d(E_k(m)) = m.$$

2.3 Gruppentheorie

Die Definitionen von Gruppen werden für die später behandelten Commitment-Verfahren benötigt. Bestimmte Arten von Gruppen besitzen Eigenschaften, welche nützlich sind bezüglich der Sicherheit von Verschlüsselungen. Zunächst werden sie allgemein definiert und danach Schrittweise spezifiziert. Diese Definitionen sind aus dem Lehrbuch von Buchmann [Buc04, Kapitel 3] entnommen, teils zusammengefasst und umgeschrieben worden.

Definition 5. *Eine* Gruppe*, notiert mit $(G, \circ)$, besteht aus einer nicht leeren Menge G mit der Verknüpfung $\circ\colon G \times G \to G$, die folgende Eigenschaften erfüllen:*

(i) $\forall a, b, c \in G\colon a \circ (b \circ c) = (a \circ b) \circ c$ *(Assoziativität)*

(ii) $\forall a \in G \exists n \in G\colon n \circ a = a$ *(neutrales Element)*

(iii) $\forall a \in G \exists i \in G\colon i \circ a = n$ *(inverses Element)*

Beispiel 3. *Die Menge der ganzen Zahlen $(..., -3, -2, -1, 0, 1, 2, 3, ...)$ mit der Verknüpfung Addition, notiert $(\mathbb{Z}, +)$, ist eine Gruppe. Assoziativität ist gegeben bei Addition. Das neutrale Element n ist $n = 0$. Jedes Element a besitzt ein inverses Element $i = -a$. Anschaulich ist -3 das inverse Element von 3, es gilt $3 + (-3) = 0 = n$. Dementsprechend ist $(\mathbb{Z}, +)$ eine Gruppe.*

Beispiel 4. *Analog ist auch die Menge der rationalen Zahlen ohne Null mit der Verknüpfung Multiplikation, notiert* $(\mathbb{Q} \setminus \{0\}, \cdot)$*, eine Gruppe. Assoziativität dank Multiplikation,* $n = 1$ *und* $i = \frac{1}{a}$*. Die Null wurde ausgelassen, da für sie kein inverses Element existiert.*

Beispiel 5. $(\mathbb{Z} \setminus \{0\}, \cdot)$ *ist keine Gruppe, da sie keine Brüche und somit nicht die benötigten inversen Elemente beinhaltet.*

Definition 6. *Die* Ordnung *einer Gruppe ist die Anzahl ihrer Elemente. Eine Gruppe heißt* endlich*, wenn ihre Ordnung nicht unendlich ist.*

Die in dieser Arbeit behandelten endlichen Gruppen besitzen immer eine obere Schranke umgesetzt durch die modulo Operation. Die additive Gruppe der ganzen Zahlen modulo m, notiert mit $(\mathbb{Z}_m^*, +)$, besteht aus den Elementen $(0, 1, 2, ..., m-1)$. Die Ordnung dieser Gruppe ist m.

Definition 7. *Sei* G *eine endliche Gruppe und* $g \in G$*. Das* Erzeugnis *von* $\langle g \rangle$ *ist* $\langle g \rangle := \{\ g^e \mid 0 \leq e < |G|\}$*. Wir nennen* $\langle g \rangle$ *Untergruppe.* G *heißt* zyklisch*, wenn* $\exists g \in G$ *mit* $\langle g \rangle = G$*. Dann nennen wir* g Generator*.*

Beispiel 6. *Trivialer Weise erzeugt* $\langle 1 \rangle$ *von* $G = (\mathbb{Z}_m^*, +)$ *wieder* G*. Somit ist* G *für alle* m *zyklisch.*
Wir erzeugen die Untergruppe $\langle 2 \rangle$ *von* $H = (\mathbb{Z}_{13}^*, \cdot)$*, um zu schauen ob* H *zyklisch ist. Dazu ergibt sich folgende Tabelle:*

e	=	0	1	2	3	4	5	6	7	8	9	10	11	12
$2^e \bmod 13$	=	1	2	4	8	3	6	12	11	9	5	10	7	1

Tabelle 2.1: Untergruppe $\langle 2 \rangle$ von H

Wir stellen fest, dass $\langle 2 \rangle$ *alle Elemente in* H *erzeugen kann und somit* H *zyklisch ist und einen Generator* $g = 2$ *besitzt. Jedoch zeigt nicht jede Untergruppe, dass* H *zyklisch ist. Die Untergruppe* $\langle 4 \rangle$ *erzeugt nur 6 Elemente aus* H*:*

e	=	0	1	2	3	4	5	6	7	8	9	10	11	12
$4^e \bmod 13$	=	1	4	3	12	9	10	1	4	3	12	9	10	1

Tabelle 2.2: Untergruppe $\langle 4 \rangle$ von H

Die eulerische Phi-Funktion $\varphi(n)$ gibt die Anzahl aller teilerfremden Zahlen von n kleiner als n an. So ist $\varphi(6) = 2$, da 1 und 5 teilerfremd zu 6 sind. Primzaheln sind nur duch sich selbst und 1 teilbar, daher gilt $\varphi(p) = p - 1$ für eine Primzahl p.

Definition 8. *Sei p eine Primzahl. Zahl a heißt* Primitivwurzel $\mod p$*, falls $\langle a \rangle$ ein Generator von $\mathbb{Z}_p^*$ ist.*
Die Gruppe $(\mathbb{Z}_p^*, \cdot)$ hat genau $p-1$ Elemente und ist zyklisch. Sie hat $\varphi(p-1)$ verschiedene Generatoren.

Primitivwurzeln bieten sich für das Verschlüsseln eines Commitments an. Denn es ist effizient zu einer Primzahl p mit einer Primitivwurzel g und einer ganzen Zahl a beispielsweise einen Commitment Schlüssel CK auszurechnen mit $CK = g^a \mod p$. Umgekehrt ist kein effizienter Algorithmus bekannt, welcher für gegebene p, g eines bekannten CK ein entsprechendes a finden kann. Diese Eigenschaft wird als die Diskrete-Logarithmus-Annahme bezeichnet [KL07, Def. 7.59].

3 Ein 1-Bit-Commitment-Verfahren

Als ein anschauliches Beispiel wird das generische 1-Bit-Commitment-Scheme aus der Arbeit von Damgård und Nielsen [DN08] eingeführt. Danach werden die möglichen Sicherheitseigenschaften bezüglich dieser Definition genauer beschrieben und die Generierung des Schlüssels behandelt.

3.1 Definition

Commitment-Verfahren: 1-Bit-Commitment

Setup: Ein probabilistischer Polynomialzeit Algorithmus G generiert mit Eingabe 1^ℓ einen öffentlichen Commitment-Schlüssel CK, wobei ℓ der Sicherheitsparameter ist. Nachdem der Schlüssel CK generiert wurde, steht er beiden Parteien zur Verfügung.

Commit: Jeder Schlüssel CK definiert eine Funktion $\texttt{commit}_{CK}\colon \{0,1\}^l \times \{0,1\} \to \{0,1\}^l$. Alice verpflichtet sich zu dem Bit b mit Commitment $C \leftarrow \texttt{commit}_{CK}(r,b)$, wobei $r \in \{0,1\}^l$ zufällig gewählt wurde. C wird Bob gesendet.

Open: Zum Öffnen von C erhält Bob r und b. Zur Korrektheit überprüft er $C = \texttt{commit}_{CK}(r,b)$.

Im Bezug auf diese Definition werden nun die Sicherheitseigenschaften erklärt.

3.2 Hiding

Die Hiding-Eigenschaft beschreibt, dass nur mit dem Commitment C der eigentliche Wert nicht bestimmt werden kann. Im Bezug zu dem „Münzwurfprotokoll“ aus der Einleitung erfüllt der Tresor diese Eigenschaft, da er nicht transparent und verriegelt ist. Die Sicherheitseigenschaft wird unterschieden je nachdem wie sehr sie von einem Commitment-Verfahren erfüllt wird. Speziell zur Definition 3.1 wird klassifiziert:

Computational Hiding: Bob, hier vertreten als ein Algorithmus in Polynomialzeit, kann nicht besser als zufällig raten, welcher Wert hinter einem Commitment C steht. Es gilt für jede zufällig erzeugte r, s $\texttt{commit}_{CK}(r,0) \sim^c \texttt{commit}_{CK}(s,1)$.

Unconditional Hiding: Bob kann trotz unbegrenzter Rechenkraft nicht besser als zufällig raten, welcher Wert hinter einem Commitment C steht. Es gilt für jede zufällig erzeugte r, s $\texttt{commit}_{CK}(r,0) \sim^s \texttt{commit}_{CK}(s,1)$.

Hierbei sollte besonders beachtet werden, dass der Wert b in der oberen Definition nicht der gleiche ist. Es sollen keine Informationen über b gewonnen werden können. Es reicht daher nicht aus, dass $\texttt{commit}_{CK}(r,0) \sim^c \texttt{commit}_{CK}(s,0)$ und/oder $\texttt{commit}_{CK}(r,1) \sim^c \texttt{commit}_{CK}(s,1)$ gelten. Dies kann zum Beispiel beides von einem Commitment-Verfahren CS_{NH} erfüllt werden, welches ein zufälliges Commitment C erzeugt wie in Abschnitt 3.1, aber zusätzlich den geheimen Wert b immer an das Ende von C schreibt. Weil der Wert b immer unverschlüsselt am Ende jedes Commitments steht, erkennt Bob dieses Muster nach dem Ausprobieren von wenigen zufällig gewählten Wertepaaren (r,b) mit der $\texttt{commit}_{CK}(r,b)$ Funktion. So kann Bob zwar nicht sagen welche Zufallsinformation r für ein erhaltenes Commitment C benutzt wurde, jedoch fällt es ihm leicht zu ermitteln ob b den Wert 0 oder 1 hat. Das Schema CS_{NH} besitzt folglich in keinster Weise die Hiding-Eigenschaft.

3.3 Binding

Die Binding-Eigenschaft bestimmt, dass es nicht möglich ist das Commitment auf verschiedene Weisen zu öffnen. Damit der Tresor im „Münzwurfprotokoll“ diese Eigenschaft erfüllt, muss nicht nur die physische Distanz zu dem Sender existieren, der Tresor darf zudem nicht unterschiedliche Zahlenkombinationen besitzen, die jeweils andere Fächer des Tresors öffnen. Genau wie die Hiding-Eigenschaft kann sie weiterhin klassifiziert werden:

Computational Binding: Alice, hier vertreten als ein Algorithmus in Polynomialzeit, ist es höchstens mit Wahrscheinlichkeit $negl(\ell)$ möglich C, r, b, r', b' zu finden bei einem Input von $G(1^\ell)$, sodass $b \neq b'$ und $\texttt{commit}_{CK}(r,b) = C = \texttt{commit}_{CK}(r',b')$.

Unconditional Binding: Alice kann trotz unbegrenzter Rechenkraft kein C, r, b, r', b' mit $b \neq b'$ und $\texttt{commit}_{CK}(r,b) = C = \texttt{commit}_{CK}(r',b')$ finden, da jedes b einzigartig durch $\texttt{commit}_{CK}(r,b)$ bestimmt ist.

Nun stellen sich folgende Fragen: Wenn die Unconditional-Varianten von Hiding und Binding eine strengere Version der Computational-Varianten sind, wozu wurden diese überhaupt definiert? Sollte nicht einfach ein Commitment-Verfahren benutzt werden, welches zur besten Sicherheit Unconditional Hiding und Unconditional Binding ist? So ein Verfahren zu benutzen ist nicht möglich.

Satz 1. *Ein Commitment-Verfahren kann nicht gleichzeitig die Eigenschaften Unconditional Hiding und Unconditional Binding erfüllen.*

Beweis 1. *Wir nehmen an, dass das 1-Bit-Commitment-Verfahren aus Abschnitt 3.1 sowohl die Unconditional-Binding-Eigenschaft, als auch die Unconditional-Hiding-Eigenschaft besitzt. Dadurch besitzt Alice theoretisch unbegrenzte Rechenkraft verschiedene Wertepaare zu finden, welche das gleiche Commitment erzeugen, und Bob besitzt theoretisch unbegrenzte Rechenkraft sein erhaltenes Commitment zu entschlüsseln, indem er alle Wertepaare ausprobiert und die erzeugten Commitments mit seinem Erhaltenem vergleicht.*

Alice möchte sich zu einem Wert $b = 0$ verpflichten. Sie rechnet $C = \texttt{commit}_{CK}(r, 0)$ aus und sendet es Bob. Es muss ein $s \neq r$ mit $C = \texttt{commit}_{CK}(s, 1)$ existieren. Ansonsten könnte Bob zweifellos sagen zu welchem Wert sich Alice mit C verpflichtet hat, weil er jedes Wertepaar (r, b) ausprobieren kann. Die Existenz von s bricht aber Unconditional Binding. Da Alice unbegrenzte Rechenkraft hat, kann sie s finden und Bob dann fälschlicher Weise sagen ihr Commitment C bestand ursprünglich aus s und $b = 1$.

3.4 Commitment-Schlüssel Generierung & Fairness

Es stellt sich noch die Frage wer genau den Commitment-Schlüssel CK generiert. Sollte dies der Sender oder Empfänger machen? Hierbei ist das Problem, dass wenn nur eine Partei den gemeinsam benutzten Commitment-Schlüssel generiert, diese den Schlüssel dann potentiell für dessen Vorteil generieren beziehungsweise manipulieren kann. So könnte Alice einen CK produzieren und an Bob schicken von dem sie weiß, dass es für ein bestimmtes Commitment mehrere valide Wertepaare gibt. Bob könnte einen CK generieren, welcher es ihm erlaubt den eigentlichen Wert eines Commitments zu sehen oder zumindest Informationen darüber auszulesen.

Zu dieser Frage gibt es mehrere Antworten. Die simpelste Lösung zu diesem Problem mit zwei misstrauischen Parteien ist es den Schlüssel nicht von Alice oder Bob generieren zu lassen, sondern von einer dritten Partei. Solange Alice und Bob der dritten Partei vertrauen, können sie unbesorgt Commitments austauschen. Das später angeschaute Pedersen Commitment-Verfahren geht in dessen Definition von dieser Methode aus, denn dadurch kann mit möglichst wenig Aufwand das Austauschverfahrens des Commitments begonnen werden. Es ist jedoch in diesem Verfahren trotzdem möglich, dass Bob den Schlüssel erzeugt und es trotzdem fair bleibt. Zumindest so fair wie es Commitment-Verfahren sein können.

Commitment-Verfahren sind immer ein wenig unfair gegenüber dem Empfänger. Dies entsteht aus der Problematik, dass der Empfänger nicht wissen kann zu welchem Wert sich der Sender verpflichtet hat, oder ob es überhaupt ein gültiges Commitment ist, bis der Sender das Commitment freigibt. Als Folge daraus könnte Alice auch einfach verweigern ihr Commitment an Bob zu beweisen. Theoretisch könnte dies auch durch eine vertraute dritte Partei gelöst werden, die vor dem Senden des Commitments an den Sender, das Commitment verifiziert. Dann wären Commitment-Verfahren keine Zwei-Partei-Protokolle mehr und verlieren die direkte Kommunikation zwischen Sender und Empfänger.

In der Praxis ist dies weniger relevant, da Commitment-Verfahren eingesetzt werden von Sendern, welche den Empfängern etwas beweisen wollen, um ein bestimmtes Ziel zu erreichen. Implizit haben deshalb Empfänger auch die Wahl den Sendern ihr Ziel zu verweigern. Sollte beispielsweise ein Nutzer das Commitment-Verfahren zur Authentifizierung bei einer Website abbrechen, wird dieser eben nicht vom System eingeloggt.

Zurück zu der eigentlichen Problematik des Schlüsselgenerierens. Wie ist es in Fällen, in denen eine solche Partei nicht existiert? Es kann tatsächlich Alice oder Bob einen

Schlüssel generieren, ohne dass der jeweils Andere Angst haben muss betrogen zu werden. Solange es möglich und effizient ist einen CK als valide zu überprüfen, kann eine Partei den Schlüssel erzeugen. Bei Commitment-Verfahren mit der Unconditional-Binding-Eigenschaft wird Alice den CK generieren. Wenn der Schlüssel validiert ist, dann hat Bob nicht zu befürchten, dass Alice mehrere Wertepaare nutzen kann. Gleichzeitig ist es dann in Alices Interesse, einen Schlüssel zu generieren, von dem sie glaubt, dass er die Hiding-Eigenschaft hinreichend erfüllt. Analog wird bei Verfahren mit der Unconditional-Hiding-Eigenschaft der Schlüssel von Bob generiert und es liegt in seiner Interesse einen guten Binding-Schlüssel zu wählen. Ein Beispiel hierfür wird im nächsten Kapitel gezeigt.

Als dritte Möglichkeit könnten sich Alice und Bob in einem Protokoll interaktiv zu einem gemeinsamen Commitment-Schlüssel einigen.

4 RSA als Commitment-Verfahren

In den Grundlagen wurde behauptet, dass Public-Key-Kryptosysteme zu Commitment-Verfahren adaptiert werden können. Dies wird in diesem Kapitel am Beispiel des Public-Key-Kryptosystems RSA gezeigt. RSA wird kurz eingeleitet und danach untersucht, inwieweit es einem Commitment-Verfahren ähnelt und ob es gegebenenfalls umzuwandeln ist. Anschließend werden allgemein kryptographische Hashfunktionen betreffend der Realisierbarkeit als Commitment-Verfahren angeschaut.

4.1 RSA Definition

Das RSA-Verfahren ist ein asymmetrisches Verschlüsselungsverfahren benannt nach seinen Erfindern Ron Rivest, Adi Shamir und Len Adleman. Es war 1971 das erste veröffentlichte Public-Key-Kryptosystem und findet heute noch weitverbreiteten Gebrauch zur verschlüsselten Kommunikation. Die Sicherheit dieses Verfahrens beruht darauf, dass noch keine effizienten Algorithmen gefunden wurden, welche die Primfaktoren von großen Zahlen ermitteln können [Buc04]. RSA ist wie folgt definiert:

Definition 9 ([Rot08, 7.1.1])**.** *RSA ist ein Public-Key-Kryptosystem mit S,E,D gegeben für die Teilnehmer Alice und Bob folgendermaßen:*

- Schlüsselerzeugung S: *Bob wählt zwei verschiedene große Primzahlen p, q mit $p \neq q$ und berechnet deren Produkt $n = pq$. Zusätzlich wählt er einen Exponent $e \in \mathbb{N}$ für den gilt:*

  $$1 < e < \varphi(n) = (p-1)(q-1) \text{ und } e, \varphi(n) \text{ sind teilerfremd.}$$

  Anschließend bestimmt er das inverse Element d von $e \mod \varphi(n)$ mit

  $$1 < d < \varphi(n) \text{ und } e \cdot d \equiv 1 \mod \varphi(n).$$

  Das Paar (n, e) ist Bobs öffentlicher Schlüssel und d sein privater Schlüssel. Bob gibt seinen öffentlichen Schlüssel für alle anderen Kommunikationsteilnehmer preis.

- Verschlüsselung E: *Alice kann nun eine Nachricht m zu einer Zahl $c = E_{n,e}(m)$ verschlüsseln, wobei die Verschlüsselungsfunktion definiert ist durch*

  $$E_{n,e}(m) = m^e \mod n.$$

  Alice sendet die verschlüsselte Zahl c an Bob.

- Entschlüsselung D: *Bob kann die erhaltene verschlüsselte Nachricht c mit seinem privaten Schlüssel d und der Entschlüsselungsfunktion*

$$D_d(c) = c^d \mod n$$

entschlüsseln.

4.2 Sicherheitseigenschaften von RSA

Commitment-Verfahren bestehen im Wesentlichen aus der Verschlüsselung eines Wertes sowie den Sicherheitseigenschaften Hiding und Binding bezüglich dieser Verschlüsselung. Solange die Verschlüsselung eines Public-Key-Kryptosystems diese Eigenschaften besitzt kann es als Commitment-Verfahren eingesetzt werden (vgl. [Gol01, 4.4.1.2]).

Binding

Die Binding-Eigenschaft übersetzt in Bezug auf generelle Kryptosysteme bedeutet, dass ein Ciphertext nur mit maximal einem einzigen bestimmten Schlüssel entschlüsselt werden darf. Die RSA-Verschlüsselung E erfüllt diese Eigenschaft ohne Änderungen. Gesendete Ciphertexte, die mit dem öffentlichen Schlüssel des Empfängers verschlüsselt wurden, können nur von dessen privaten Schlüssel entschlüsselt werden. Da jeder öffentliche Schlüssel einen einzigartigen privaten Schlüssel bestimmt, gibt es nur diesen einen Weg den Ciphertext zu entschlüsseln. Somit gilt die Unconditional-Binding-Eigenschaft.

Hiding

Für Public-Key-Kryptosysteme existiert bereits ein Analogon zu der Hiding-Eigenschaft der Commitment-Verfahren. Sollte ein Public-Key-Kryptosystem das „Chosen Plaintext Attack indistinguishability experiment“, kurz IND-CPA, bestehen, gilt die Computational-Hiding-Eigenschaft von Commitment-Verfahren. Dieses Experiment besteht aus einem Angriff, bei dem ein Angreifer zwei Klartexte besitzt und zufällig einen davon verschlüsselt zurückbekommt. Der Angreifer kennt das Verschlüsselungsverfahren sowie öffentliche Informationen, wie den öffentlichen Schlüssel, und ist beschränkt in probabilistischer Polynomialzeit. Kann der Angreifer nur mit einer vernachlässigbaren Wahrscheinlichkeit öfter als 50% bestimmen, welcher seiner beiden Klartexte verschlüsselt wurde, gilt das Kryptosystem als IND-CPA sicher (vlg. [KL07, 10.4]). Mit anderen Worten ist das Ergebnis der Verschlüsselung für den Polynomialzeit beschränkten Angreifer ununterscheidbar von der Eingabe, was äquivalent zu der Definition von Computational Hiding im Rahmen der Commitment-Verfahren ist.

Satz 2. *Die obige RSA-Definition, auch „Textbook-RSA“ genannt, ist deterministisch und damit nicht IND-CPA sicher.*

Beweis 2. *Stellt man sich solch einen Angriff auf Textbook-RSA vor, wird der Grund sofort klar. Der Angreifer kann lediglich beide Klartexte verschlüsseln und sie mit dem erhaltenen Ciphertext vergleichen, weil er Zugriff auf den öffentlichen Schlüssel und die Verschlüsselungsmethode hat. Dadurch weiß der Angreifer welcher Klartext verschlüsselt wurde und liegt beim „Raten“ immer korrekt.*

Aus diesem Grund wird diese Art des RSA heutzutage nicht mehr eingesetzt. Stattdessen werden Versionen von RSA benutzt, welche durch zufälliges Padding des Klartexts die Verschlüsselung nicht mehr deterministisch machen und somit IND-CPA sicher wird. Zurückerinnert an die Grundlagen hat das Hinzufügen von zusätzlichen Zufallsinformation an die Nachricht in Commitments den gleichen Hintergrund. Ein Beispiel für RSA mit zufälligem Padding ist RSA-OAEP, das bewiesen IND-CPA sicher ist [BR94].

4.3 Benutzung als Commitment-Verfahren

Durch die Erfüllung der Sicherheitseigenschaften von RSA mit zufälligem Padding kann es als Commitment-Verfahren benutzt werden. Eine entsprechende Umsetzung kann so aussehen:

Commitment-Verfahren: RSA

Setup: Alice generiert wie in Definition 9 den öffentlichen Schlüssel $CK = (n, e)$ und den privaten Schlüssel d. Sie sendet den öffentlichen Schlüssel an Bob. Alice generiert die Schlüssel, denn wie bereits besprochen besitzt RSA die Unconditional-Binding-Eigenschaft und Bob besitzt die Möglichkeit seinen erhaltenen öffentlichen Schlüssel CK zu überprüfen. Zusätzlich haben sich Alice und Bob auf eine Funktion $\circ : m \circ r = m'$ geeinigt, welche einer Nachricht Zufallsinformationen hinzufügt.

Commit: Alice legt sich nun auf eine Zahl m fest. Dieser Zahl wird eine Zufallszahl r hinzugefügt und so zu einem $m \circ r = m'$ modifiziert, da sonst die Hiding-Eigenschaft nicht mehr gilt. Sie berechnet das Commitment $C = E_{n,e}(m')$ und sendet dies an Bob.

Open: Bob kann nicht vorzeitig C öffnen. Dazu müsste er in der Lage sein die RSA-Verschlüsselung brechen können, wozu noch kein effizienter Algorithmus gefunden wurde. Alice enthüllt ihre Zahl m und Zufallszahl r und Bob kontrolliert nun, ob $m \circ r = m'$ und $C = E_{n,e}(m')$.

In dieser Umsetzung fällt auf, dass der private Schlüssel von Alice nie benutzt wurde. Dies unterstreicht den Unterschied zwischen Commitment-Verfahren und Kryptosystemen. Kryptosysteme beinhalten zu der Verschlüsselung einer Nachricht auch die Fähigkeit Ciphertexte mit einem bestimmten Schlüssel wieder zu entschlüsseln. Commitment-Verfahren illustrieren lediglich ihre Fähigkeit des „Verschlüsselns“ anhand ihres Klartext abhängigen „Schlüssels“.

Der Aufbau von Commitment-Verfahren ist einfacher gestaltet als der von Kryptosystemen. Public-Key-Kryptosysteme besitzen Eigenschaften, die ein Commitment-Verfahren nicht benötigt und sich bei einer Wiederverwendung als Commitment-Verfahren negativ bemerkbar machen können. So gibt es bei der Verwendung von Public-Key-Kryptosystemen die zusätzliche Gefahr den privaten Schlüssel aus dem Öffentlichen ableiten zu können. Sollte beispielsweise ein effizienter Weg gefunden werden einen privaten RSA-Schlüssel zu dem öffentlichen RSA-Schlüssel zu finden, kann ein Angreifer in dieser Umsetzung das Commitment C einfach entschlüsseln, dadurch m' erhalten und somit potentiell Rückschlüsse auf m ziehen.

Beispiel 7. *In diesem Beispiel wird die Benutzung des Commitment-Verfahrens basierend auf RSA mit konkreten Zahlenwerten gezeigt. Zur besseren Lesbarkeit und Verständnis werden kleine Zahlen und kurzes Zufallsmaterial benutzt. In der Praxis müssten diese weitaus größer beziehungsweise länger gewählt werden.*

Setup *Alice möchte ein Commitment an Bob schicken. Zur Erzeugung des öffentlichen Schlüssels wählt sie die Primzahlen* $p = 17$ *und* $q = 29$ *und rechnet das Produkt* $n = p \cdot q = 17 \cdot 29 = 493$ *aus. Nun muss sie noch einen Exponenten* e *wählen für den gilt* $1 < e < \varphi(493) = 448$*. Zusätzlich müssen* e *und* 448 *teilerfremd sein. Sie wählt* $e = 3$*. Der öffentliche Schlüssel ist also* $CK = (493, 3)$*. Den privaten Schlüssel muss sie nicht erzeugen, weil dieser nicht gebraucht wird.*

Als Methode den Klartext mit Zufallsinformation zu bereichern, einigen sich Alice und Bob darauf, eine Zahl m *in Binärdarstellung mit einer Bitfolge* r *zu verketten und die resultierende Zahl wieder in Dezimaldarstellung umzuwandeln, geschrieben als* $m||r = m'$*.*

Commit *Nun kann Alice ein Commitment erzeugen und dies an Bob senden. Alice legt sich auf die Zahl* $m = 2$ *fest. Als zufällige Bitfolge generiert sie* $r = 1011$*. Sie berechnet*

$$m' = 2||1011 = 101011 = 43.$$

Mit m' *kann sie nun das Commitment*

$$C = E_{493,3}(43) = 43^3 \pmod{493} \equiv 134 \pmod{493}$$

erzeugen. Alice schickt $C = 134$ *und* CK *an Bob.*

Bob könnte in diesem konkreten Beispiel die Primzahlen p, q *aufgrund der Kürze von* 493 *erraten. Die Computational-Hiding-Eigenschaft würde dann nicht mehr gelten. In der Praxis würde es ihm nicht so leicht fallen, da öffentliche RSA-Schlüssel* n *mit einer Länge von mindestens 2048 Bit enthalten sollten laut dem National Institute of Standards and Technology aus den USA (vlg. [BD06]). Damit auch die Unconditional-Binding-Eigenschaft gilt, muss* $m' < n$ *gelten, da es unendlich viele Lösungen für* m' *der Form* $493 \cdot t + 43 \equiv 134 \pmod{493}, t \in \mathbb{N}$ *gibt.*

Open *Wenn Alice bereit ist ihr Commitment zu offenbaren schickt sie* $m = 3$ *und* $r = 1011$ *an Bob. Nun kann Bob überprüfen, ob* $43 = 3||1011$ *und* $134 \equiv 43^3 \pmod{493}$. *Wenn beides gilt, ist das Verfahren erfolgreich beendet.*

4.4 Kryptographische Hashfunktionen als Commitment-Verfahren

Nach dem Behandeln des Potentials von Public-Key-Kryptosystemen als Commitment-Verfahren stellen sich Leser mit Erfahrung im Bereich Kryptographie vermutlich die Frage, ob Hashfunktionen ebenso geeignet sein könnten. Diese bilden einen bestimmten Wert zu einem Hashwert ab, ohne die Option den eigentlichen Wert von dem Hashwert zu rekonstruieren. Prinzipiell gleicht dies der Commit-Phase eines Commiment-Verfahrens. Hashfunktionen finden beispielsweise Benutzung bei dem Speichern von Passwörtern in Datenbanken, welche Nutzerdaten zur Authentifizierung enthalten. So erhalten Angreifer bei einem erfolgreichen Angriff auf eine Datenbank nicht die Passwörter der Nutzer als Klartext. Die folgenden Definitionen sind zusammengefasst aus Kapitel 12 des Lehrbuchs von Buchmann [Buc04].

Definition 10. *Eine* Hashfunktion *ist die Abbildung*

$$h : \Sigma^* \to \Sigma^n,\ n \in \mathbb{N}, a \mapsto o$$

mit den folgenden Eigenschaften:

- *Die Eingabe* a *aus dem Alphabet* Σ *darf beliebig groß gewählt werden.*
- *Die Ausgabe* o *aus dem Alphabet* Σ*, in dieser Arbeit auch Hashwert genannt, besitzt immer die feste Länge* n*.*
- *Die Hashfunktion* h *ist effizient berechenbar.*

Eine kryptographische Hashfunktion *ist* kollisionsresistent *und* hiding.

Definition 11. *Eine Hashfunktion* h *ist* kollisionsresistent*, wenn es unmöglich erscheint Eingaben* a *und* b *zu finden, sodass* $a \neq b$ *und* $h(a) = h(b)$ *gilt.*

Definition 12. *Eine Hashfunktion gilt als* hiding*, wenn es keinen effizienten Weg gibt, für eine Ausgabe* $o = h(a)$ *den Eingabewert* a *zu finden.*

Eine beliebige kryptographische Hashfunktion als Commiment-Verfahren zu benutzen würde in etwa wie folgt aussehen:

Commitment-Verfahren: Kryptographische Hashfunktion

Setup: Als öffentlicher Parameter existiert nur die Hashfunktion h, auf die Alice und Bob Zugriff haben.

Commit: Alice möchte sich zu einer Nachricht m verpflichten und generiert dazu ein zufälliges r. Sie berechnet das Commitment $C = h(m||r)$ und schickt es an Bob.

Open: Bob besitzt C und nachdem Alice bereit ist ihren Wert m aufzudecken schickt sie m, r an Bob. Nun kann er überprüfen, ob $C = h(m||r)$ gilt.

Auffällig ist, dass in dieser Ausführung die Problematik der Generierung des öffentlichen Schlüssel wegfällt, weil der öffentliche Schlüssel gar nicht existiert. Die Sicherheit dieses Verfahrens beruht nur auf den Eigenschaften der verwendeten Hashfunktion. Angriffe müssen sich bei herkömmlichen Commitment-Verfahren auf den jeweils einzigartigen öffentlichen Schlüssel pro Durchführung in Verbindung mit der Commit-Funktion konzentrieren. Ausführungen basierend auf Hashfunktionen können nur verwendet werden, wenn deren Sicherheit vor Angreifern im Kontext von Commitments ausreichend bewiesen ist.

Sicherheit

Generell sollte klar sein, dass Hashfunktionen nie die Unconditional-Binding-Eigenschaft besitzen können. Durch die Kollisionsresistenz von kryptographischen Hashfunktionen sind Kollisionen zwar schwer zu finden, müssen aber zwingend existieren, weil die Menge aller Ausgaben kleiner ist als die Menge aller Eingaben. Ein garantierter Weg eine Kollision zu finden, bei einer Hashfunktion mit n möglichen Ausgaben, ist es $n + 1$ Hashwerte zu berechnen, so wird mindestens eine Kollision gefunden. Auf Grund der großen Menge an möglichen Ausgaben ist dieser Ansatz bei kryptographischen Hashfunktionen zu naiv. Mit unbegrenzter Rechenkraft könnte Alice jedoch folglich m', r' finden, sodass gilt $m \neq m', r \neq r', C = h(m||r) = h(m'||r')$. Da solche Kollisionen aber aufgrund der Kollisionsresistenz nicht effizient auffindbar sind, besitzen Umsetzungen von Commitment-Verfahren durch Hash-Funktionen zumindest die Computational-Binding-Eigenschaft.

Angriffe auf die Hiding-Eigenschaft von Hashfunktionen finden beispielsweise mit zuvor berechneten Hashmaps, welche die Werte und deren Hashwerte beinhaltet, statt. In Bezug auf die obige Nutzung von Hashes zur Passwortspeicherung in Datenbanken, können Angreifer beispielsweise Hashmaps mit den 1000 meist verwendeten Passwörtern einsetzen und so hoffen zumindest einige Anmeldedaten von Nutzern zu erhalten. Aus diesem Grund sollte man den Wert vor dem Hashen mit einem sogenannten „Salt“ randomisieren $h(wert||salt)$, bevor man ihn in die Datenbank speichert. Da die Hiding-Eigenschaft bei der Benutzung als Commitment-Verfahren ebenso angreifbar ist, wurde

der Salt in Form von dem zufällig gewählten r bei dieser Ausführung berücksichtigt. Folgendermaßen gilt also die Computational-Hiding-Eigenschaft.

Es wird weiter untersucht, ob die Unconditional-Hiding-Eigenschaft gelten könnte. Für Bob bedeutet Unconditional Hiding, dass obwohl er alle möglichen Commitments berechnet hat, sich nicht sicher sein kann zu welchen Werten sich Alice verpflichtet hat. Aufgrund der Existenz von Wertepaaren, welche das gleiche Commitment erzeugen. In diesem Fall müssten Klartexte m, m' und Zufallswerte r, r' existieren für die gilt $m \neq m', r \neq r', h(m||r) = h(m'||r')$. Hashfunktionen garantieren diese Eigenschaft nicht. Potentiell könnten Hashwerte existieren, die nur durch eine bestimmte Eingabe erzeugt werden können. Folglich gilt die Unconditional-Hiding-Eigenschaft nicht.

5 Pedersen Commitment-Verfahren

Das Pedersen Commitment-Verfahren ist eines der bekanntesten Commitment-Verfahren mit der Unconditional-Hiding-Eigenschaft und wird deshalb hier thematisiert. Zuerst wird das Verfahren definiert und dessen Sicherheitseigenschaften untersucht. Darauf folgt ein anschauliches Beispiel zur Benutzung des Verfahrens. Zum Abschluss wird auf die homomorphe Eigenschaft hingewiesen und dargestellt, wozu sie benutzt werden kann.

5.1 Definition

Dieses Commitment-Verfahren wurde 1991 von Torben Pryds Pedersen auf einer Kryptographiekonferenz vorgestellt [Ped91]. Die Sicherheit dieses Verfahrens beruht auf der Diskreten-Logarithmus-Annahme.

Commitment-Verfahren: Pedersen

Setup: Gegeben sind zwei große Primzahlen p und q, sodass gilt q teilt $p-1$. Sei G_q Teilmenge der zyklischen Gruppe $\mathbb{Z}_p^*$ der Ordnung q und g ein Generator von G_q. Eine dritte Partei wählt $g, h \in G_q$, sodass niemand den diskreten Logarithmus h in Bezug auf g kennt, also kein x bekannt ist, sodass $h = g^x \pmod p$.

Commit: Um sich einem $m \in \mathbb{Z}_q$ zu verpflichten, wählt Alice zufällig ein $r \in \mathbb{Z}_q$ und berechnet das Commitment $C = \texttt{commit}(m, r) = g^m h^r$. Sie schickt C an Bob.

Open: Damit Bob das Commitment $C \in G_q$ öffnen kann, enthüllt Alice ihre Werte m, r. Bob berechnet selber das Commitment $C' = \texttt{commit}(m, r) = g^m h^r$ und akzeptiert solange $C = C'$.

5.2 Sicherheitseigenschaften

Binding

Um die Binding-Eigenschaft zu brechen müsste Alice verschiedene Nachrichten m, m' finden, die das gleiche Commitment C erzeugen. Es müsste $g^m h^r = g^{m'} h^{r'}$ gelten. Um m', r' zu finden müsste sie den Logarithmus von h zur Basis g kennen: $log_g(h) = \dfrac{(m - m')}{(r' - r)}$. Wenn die Diskrete-Algorithmus-Annahme gilt, ist dieser Logarithmus nicht effizient zu berechnen und somit für eine Polynomialzeit begrenzte Alice nicht auffindbar. Folglich gilt die Computational-Binding-Eigenschaft. Wäre Alice nicht

begrenzt in Rechenkraft, kann sie $log_g(h)$ finden, deshalb besitzt das Verfahren nicht die Unconditional-Binding-Eigenschaft.

Hiding

In diesem Verfahren gilt die Unconditinal-Hiding-Eigenschaft. Falls Bob passende m, r zu einem Commitment findet, kann er nicht beurteilen, ob dies die m, r sind, die Alice für ihr Commitment benutzt hat. Jedes m ist aus Bobs Perspektive gleich wahrscheinlich verpflichtet worden zu sein als C, solange r zufällig gewählt wurde. Wenn m, r und eine unterschiedliche Nachricht m' gegeben sind, existiert ein r', sodass $g^m h^r = g^{m'} h^{r'}$ gilt. Wir können $log_g(h) = \frac{(m-m')}{(r'-r)}$ für r' umstellen: $r' = \frac{(m-m')}{log_g(h)} + r$.

Da für Bob die Kenntnis über $log_g(h)$ nicht nützlich ist, aber Alice damit die Binding-Eigenschaft brechen könnte, generiert er die öffentlichen Faktoren des Commitment-Verfahrens. So kann er mit einem geheimen a ein $h = g^a \mod p$ generieren, von dem er selbst überzeugt ist, dass Alice a nicht berechnen kann.

5.3 Benutzung

Die Nutzung des Pedersen Commitment-Verfahren wird hier mit konkreten Zahlenwerten gezeigt. Die in diesem Beispiel benutzten Primzahlen werden im Vergleich zur wirklichen Anwendung sehr klein gewählt, zur besseren Übersicht und Fähigkeit die Ergebnisse bei Bedarf selbst nachzurechnen. Obwohl das Prinzip gleich bleibt, sollten in der praktischen Nutzung des Verfahrens die Primzahlen groß genug gewählt werden, sodass die Diskrete-Logrithmus-Annahme gilt und somit auch die Computational-Binding-Eigenschaft.

Setup Bob generiert die öffentlichen Parameter, weil das Verfahren Unconditional Hiding ist und Alice die Parameter überprüfen kann. Bob wählt die Primzahlen $p = 479$ und $q = 239$, sodass gilt q teilt $p - 1$. Dies kann er beispielsweise errechnen, in dem er eine Primzahl q wählt und $p = i \cdot q + 1$ für verschiedene $2 \leq i \in \mathbb{N}$ errechnet, bis p eine Primzahl ist.

Es fehlen noch die Elemente g, h aus der Untergruppe G_q von $\mathbb{Z}_p^*$, wobei g ein Generator für G_q ist. Stichprobenartig kann Bob Elemente $n \in \mathbb{Z}_p^*$ nehmen und prüfen, ob es in G_q ist, da gilt:

$$a \in G_q \iff n^q \equiv 1 \pmod{p}$$

Bob entscheidet sich für den Generator $g = 125$. Er wählt zufällig den geheimen Schlüssel $a = 206$ aus dem Intervall $[1 : q - 1]$. Damit errechnet er

$$\begin{aligned} h &= g^a \pmod{p} \\ &= 125^{206} \pmod{479} \\ &= 324 \end{aligned}$$

Damit sind alle öffentlichen Parameter gesetzt und Bob sendet Alice $p = 479, q = 239, g = 125, h = 324$. Alice überprüft nun, ob p, q Primzahlen sind. Zusätzlich noch, ob $p - 1$ ein Vielfaches von q ist, beispielsweise mit

$$p \bmod q = 1 \Rightarrow 479 \bmod 239 = 1$$

Anschließend überprüft sie, ob g, h Elemente aus G_q sind:

$$\begin{aligned} n^q \equiv 1 \pmod{p} &\Rightarrow 125^{239} \equiv 1 \pmod{p} \\ &\Rightarrow 324^{239} \equiv 1 \pmod{p} \end{aligned}$$

Wenn Alice die Parameter erfolgreich überprüft hat, akzeptiert sie diese.

Commit Alice wählt m und generiert r aus $\mathbb{Z}_q$. Sie verpflichtet sich zu der Nachricht $m = 27$ mit der generierten Zufallszahl $r = 55$. Sie berechnet dafür das Commitment $C = c(m, r)$:

$$\begin{aligned} c(m, r) &= g^m \cdot h^r \pmod{p} \\ c(27, 55) &= 125^{27} \cdot 324^{55} \pmod{479} \\ &= 366 \end{aligned}$$

Alice schickt ihr Commitment $C = 366$ an Bob.

Open Nach einiger Zeit gilt es für Alice ihr Commitment zu öffnen. Alice ist jedoch aufgefallen, dass sie sich viel lieber zu $m = 28$ anstatt $m = 27$ verpflichtet hätte. Naiver Weise probiert sie Bob zu täuschen, indem sie $m = 28$ und $r = 55$ zur Bestätigung sendet. Bob überprüft, ob Alice die Wahrheit sagt. Er errechnet daher das Commitment $C' = c(28, 55)$:

$$\begin{aligned} c(m, r) &= g^m \cdot h^r \pmod{p} \\ c(28, 55) &= 125^{28} \cdot 324^{55} \pmod{479} \\ &= 245 \end{aligned}$$

Bob akzeptiert nicht, weil $C' = 245 \neq 366 = C$ ist. Dies wäre Grund genug das Verfahren abzubrechen und zu beenden. Stattdessen fordert er Alice auf ihre wahren Werte zu senden, um das Verfahren erfolgreich abzuschließen. Alice gibt nach und sendet $m = 27, r = 55$. Bob überprüft erneut errechnet ein C'':

$$\begin{aligned} c(m, r) &= g^m \cdot h^r \pmod{p} \\ c(27, 55) &= 125^{27} \cdot 324^{55} \pmod{479} \\ &= 366 \end{aligned}$$

Diesmal akzeptiert Bob, weil $C'' = 366 = C$ ist. Damit ist das Verfahren beendet.

5.4 Homomorphie

Das Pedersen Commitment-Verfahren besitzt nicht nur die Unconditional-Hiding-Eigenschaft, es ist zudem additiv homomorph.

Definition 13 ([Buc04, 3.16.3])**.** *Es seien die Gruppen $(X, \circ), (Y, \bullet)$ gegeben. Eine Abbildung $f : X \to Y$ heißt* Homomorphismus*, wenn für $f(a \circ b) = f(a) \bullet f(b)$ gilt für alle $a, b \in X$ und $\circ, \bullet$ sind Verknüpfungen.*

Beispiel 8. *Die Funktion $f(x) = e^x$ ist ein Homomorphismus der Gruppe $(\mathbb{R}^*, \cdot)$ zu $(\mathbb{R}^*, +)$, denn es gilt:*

$$f(a) \cdot f(b) = e^a \cdot e^b = e^{a+b} = f(a + b)$$

Sehr ähnlich zu diesem Beispiel ist auch das Pedersen Commitment-Verfahren homomorph, in $\mathbb{Z}_p^*$ gilt:

$$\begin{aligned} c(m_1, r_1) \cdot c(m_2, r_2) &\equiv (g^{m_1} \cdot h^{r_1}) \cdot (g^{m_2} \cdot h^{r_2}) \\ &\equiv (g^{m_1} \cdot g^{m_2}) \cdot (h^{r_1} \cdot h^{r_2}) \\ &\equiv g^{m_1+m_2} \cdot h^{r_1+r_2} \\ &\equiv c(m_1 + m_2, r_1 + r_2) \end{aligned}$$

Diese Eigenschaft erlaubt es Alice, dass Bob nur die Summe ihrer als Commitment verpflichteten Werte erfährt. Sollten Bob drei Commitments von Alice interessieren, muss er nicht jedes einzelne Commitment öffnen, sondern kann deren Produkt berechnen und dies mit Alices Hilfe öffnen. Es folgt ein Beispiel, welches nur durch die additive Homomorphie des Verfahrens ermöglicht werden kann.

5.5 Möglicher Anwendungsbereich

Ein Staat besitzt ein simples Steuermodell. Monatlich müssen alle Bürger 20 Prozent ihrer Ausgaben als Steuer bezahlen. Zusätzlich ist dem Staat die Privatsphäre seiner Bürger wichtig. So soll er nicht Zugriff auf jede Transaktion eines Bürgers haben, sondern die Steuern nur anhand der Summe der Transaktionen am Ende eines Monats erheben. Weil der Staat aber nur das Ergebnis der Ausgaben seiner Bürger erfährt, haben diese die Möglichkeit zu betrügen und Steuern zu hinterziehen. Dieses Problem könnte durch das Petersen Commitment-Verfahren dank der additiven Homomorphie gelöst werden.

Der Staat verteilt jeden Monat an jeden Bürger einzeln die öffentlichen Parameter zur Erzeugung von Commitments. Eine Bürgerin Alice gibt in einem Monat in $n \in \mathbb{N}$ Transaktionen Geld aus. Pro Transaktion berechnet sie ein Commitment $C = g^m h^r$, wobei m der Geldbetrag und r das Zufallsmaterial der Transaktion sind. Sie speichert m, r privat und sendet C an den Staat. Am Ende des Monats möchte der Staat die Steuern einziehen, besitzt aber nur n Commitments und fordert deshalb Alice auf die

Summe ihrer Ausgaben freizugeben. Alice berechnet die Summe M ihrer Ausgaben und die Summe R des Zufallsmaterials:

$$M = \sum_{i=1}^{n} m_i \ , \ R = \sum_{i=1}^{n} r_i$$

Sie sendet M und R an den Staat um fortzufahren. Der Staat berechnet nun das Commitment der Summe C_{sum}, indem er das Produkt der in diesem Monat von Alice erhaltenen Commitments bildet:

$$C_{sum} = \prod_{i=1}^{n} C_i = \prod_{i=1}^{n} g^{m_i} h^{r_i} = g^{m_{sum}} h^{r_{sum}}, \text{ mit } m_{sum} = \sum_{i=1}^{n} m_i, \ r_{sum} \text{ analog.}$$

Zur Korrektheit der von Alice Übertragenen Summe M überprüft er, ob $C_{sum} = g^M h^R$. Wenn dies gilt kann sich der Staat sicher sein, dass Alice nicht betrogen hat und angemessene Steuern erheben.

In einer ähnlichen Rolle wird das Pedersen Commitment-Verfahren als Grundlage für Confidential Transactions bei Kryptowährung wie Bitcoin eingesetzt (siehe [Ada]).

6 ElGamal Commitment-Verfahren

Das ElGamal Commitment-Verfahren wird hier vorgestellt, da es dem Pedersen Commitment-Verfahren ähnelt, es jedoch das Gegenstück im Bezug zu den Sicherheitseigenschaften Hiding und Binding bildet. Es wird das grundlegende ElGamal Public-Key-Kryptosystem und dessen Nutzung als Commiment-Verfahren definiert. Wie üblich werden danach die Sicherheitseigenschaften behandelt und danach ein mögliches Szenario beschrieben, in dem das Verfahren benutzt werden kann.

6.1 Definition

Taher A. Elgamal stellte 1985 das ElGamal Public-Key-Kryptosystem vor, dessen Sicherheit auf der Diskreten-Logarithmus-Annahme beruht [Gam85]. Das ElGamal Commitment-Verfahren leitet sich aus diesem Kryptosystem ab. Genauer gesagt ist es einer Abwandlung des sogenannten „exponential ElGamal"Kryptosystems, welches als Commitment-Verfahren eingesetzt wird.

Definition 14 ([Buc04, 9.6],[BW13])**.** *Das ElGamal Public-Key-Kryptosystem ist durch S,E,D gegeben mit:*

- Schlüsselerzeugung S: *Gegeben sind zwei große Primzahlen p und q, sodass gilt q teilt $p-1$. Sei G_q Teilmenge der zyklischen Gruppe $\mathbb{Z}_p^*$ der Ordnung q und g ein Generator von G_q. Man wählt zufällig einen privaten Schlüssel $a \in \mathbb{Z}_q$, der öffentliche Schlüssel h errechnet sich durch $h = g^a$.*
- Verschlüsselung E: *Um eine Nachricht $m \in \mathbb{Z}_q$ zu verschlüsseln, wird ein zufälliges $r \in \mathbb{Z}_q$ gewählt. Der Ciphertext (c_1, c_2) ist gegeben mit:*

$$(c_1, c_2) = (g^r, g^m h^r)$$

- Entschlüsselung D: *Um einen Ciphertext (c_1, c_2) zu entschlüsseln, wird berechnet:*

$$g^m = \frac{c_2}{(c_1)^a}$$

Hierbei anzumerken ist, dass es sich bei dieser Definition nicht um das normale ElGamal Public-Key-Kryptosystem handelt, sondern um die exponentielle ElGamal Variante. Normalerweise wird $c_2 = mh^r$ anstatt $c_2 = g^m h^r$ berechnet und der Ciphertext wird zu einem m entschlüsselt anstatt zu g^m. Zum Einsatz als Kryptosystem scheint diese Änderung suboptimal, weil dadurch ein weiterer Schritt benötigt wird m zu finden. Allerdings ermöglicht sie, dass das Kryptosystem additiv homomorph wird (vgl. [BW13]).

Als Commitment-Verfahren ist das ElGamal Public-Key-Kryptosystem so anzuwenden:

Commitment-Verfahren: ElGamal

Setup: Gleicht sich mit der Schlüsselerzeugung S des Public-Key-Kryptosystems.

Commit: Um sich einem $m \in \mathbb{Z}_q$ zu verpflichten, wählt Alice zufällig ein $r \in \mathbb{Z}_q$ und berechnet das Commitment $C = \texttt{commit}(m, r) = (c_1 = g^r, c_2 = g^m h^r)$.

Open: Damit Bob das Commitment $C = (c_1, c_2) \in G_q$ öffnen kann, enthüllt Alice m, r. Bob berechnet selber das Commitment $C' = \texttt{commit}(m, r) = (g^r, g^m h^r)$ und akzeptiert, solange $C = C'$.

6.2 Sicherheitseigenschaften

Binding

Das ElGamal Commitment-Verfahren ist sehr ähnlich zu dem Pedersen Commitment-Verfahren. Würde das c_1 aus dem Commitment C weglassen werden, wäre es ein Pedersen Commitment. Daraus wird klar, weshalb dieses Commitment-Verfahren auch additiv homomorph ist. Allerdings bewirkt das Beifügen von c_1 an das Commitment die Unconditional-Binding-Eigenschaft, weil es r fixiert.

Alice kann kein $r \neq r'$ finden, sodass $g^r = g^{r'}$ gilt. Weil r aus der Gruppe $\mathbb{Z}_q$ stammt gilt, dass $g^r \neq g^{r'}$ aus $r \neq r'$ folgt. Wie aus den Grundlagen bekannt ist, erzeugt ein Generator wie g für eine zyklische Gruppe einer Ordnung x zusammen mit einem Exponenten $y \in [1 : x - 1]$ alle Elemente der Gruppe einmalig. Aus diesem Grund gilt die Unconditional-Binding-Eigenschaft.

Hiding

Das Verfahren kann nicht Unconditional Hiding sein, weil es schon Unconditional Binding ist. Es wird untersucht, ob die Computational-Hiding-Eigenschaft erfüllt ist. Der erste Angriffspunkt für Bob wäre r zu finden mit $log_g(c1) = r$. Würde er dies schaffen kann er $c_2 = g^m h^r$ umstellen und folglich m berechnen mit $log_g(\frac{c_2}{h^r}) = m$. Unter der Diskreten-Logrithmus-Annahme sind diese Logarithmen nicht effizient berechenbar. Die Computational-Hiding-Eigenschaft ist damit erfüllt.

6.3 Benutzung

Dieses Beispiel wird nochmals die Ähnlichkeit zu dem Pedersen Commitment-Verfahren betonen. Es gilt dasselbe bezüglich der Wahl der öffentlichen Parameter zur Sicherheit, wie in dem Unterkapitel 5.3 der Benutzung des Pedersen Commitment-Verfahrens.

In einer echten Anwendung des Verfahrens sollten die Primzahlen groß genug gewählt werden, sodass die Diskrete-Logarithmus-Annahme gilt und somit auch die Computational-Hiding-Eigenschaft des ElGamal Commitment-Verfahrens.

Setup Alice erzeugt die öffentlichen Parameter, weil das Verfahren Unconditional Binding ist und Bob die Parameter überprüfen kann. Alice erzeugt diese Parameter auf dieselbe Weise wie Bob in 5.3. Damit die Beispiele nicht zu gleich werden, werden hier andere Zahlenwerte für die Parameter benutzt. Alice wählt folgende Parameter:

$$p = 1627, q = 271, g = 1195, a = 104, h = 964$$

und sendet p, q, g, h an Bob. Bob überprüft nun die Parameter, wie Alice es am Ende des Setups in 5.3 gemacht hat. Bob akzeptiert, wenn alle Parameter seine Überprüfung bestanden haben.

Commit Alice wählt m und generiert r aus $\mathbb{Z}_q$. Sie verpflichtet sich zu der Nachricht $m = 72$ mit der generierten Zufallszahl $r = 44$. Sie berechnet dafür das Commitment $C = (c_1, c_2) = c(m, r)$:

$$\begin{aligned} c(m, r) &= (g^r, g^m \cdot h^r) \pmod{p} \\ c(72, 44) &= (271^{44}, 271^{72} \cdot 964^{44}) \pmod{1627} \\ &= (945, 1385) \end{aligned}$$

Alice schickt ihr Commitment $C = (945, 1385)$ an Bob.

Open Nach einiger Zeit gilt es für Alice, ihr Commitment zu öffnen. Alice schickt daher ihre Werte $m = 72, r = 44$ an Bob. Zur Überprüfung errechnet Bob das Commitment C':

$$\begin{aligned} c(m, r) &= (g^r, g^m \cdot h^r) \pmod{p} \\ c(72, 44) &= (271^{44}, 271^{72} \cdot 964^{44}) \pmod{1627} \\ &= (945, 1385) \end{aligned}$$

Bob akzeptiert, weil $C' = (945, 1385) = C$ ist. Damit ist das Verfahren beendet.

6.4 Möglicher Anwendungsbereich

Ein Sportverein möchte darüber entscheiden, ob neue Geräte gekauft werden sollen oder nicht. Die Mitglieder des Vereins wollen diese Entscheidung durch eine demokratische Wahl treffen, weil der Kauf neuer Geräte den Mitgliedsbeitrag steigern würde. Diese Wahl besitzt die Regeln: Jedes Vereinsmitglied ist wahlberechtigt und kann mit „Ja“ oder „Nein“ abstimmen, alle Wähler sind an ihre Wahl gebunden, die Wahl soll geheim bleiben und das Ergebnis für jedes Mitglied nachvollziehbar sein. Der Verein hat sich folgendes Wahlprinzip ausgedacht:

Die Vereinsvorsitzende T stellt die öffentlichen Parameter, die für das ElGamal Commitment-Verfahren und Kryptosystem nötig sind, weil alle Mitglieder ihr vertrauen. Der Verein besitzt n Mitglieder beziehungsweise Wähler $W_1, ..., W_n$. Um eine Stimme abzugeben, wählt ein Wähler W_i eine Nachricht $m_i \in 0, 1$ für „Nein/Ja“, ein zufälliges $r_i \in \mathbb{Z}_q$ und berechnet das Commitment $C_i = (g^{r_i}, g^{m_i} h^{r_i})$. Jedes C_i wird öffentlich gemacht und jeder Wähler sendet r_i an T. Die Vereinsvorsitzende T berechnet dann:

$$r_{sum} = \sum_{i=1}^{n} r_i$$

und veröffentlicht r_{sum}.

Jedes Mitglied errechnet C_{sum} mit:

$$\prod_{i=1}^{n} C_i = (g^{r_{sum}}, g^{m_{sum}} h^{r_{sum}})$$

Weil nun $g^{r_{sum}}$ und $h^{r_{sum}}$ bekannt sind, erzeugen sie Commitments mit r_{sum} und verschiedenen $m' \in [0 : n]$, bis ein Commitment $C(m', r_{sum})$ gleich C_{sum} gefunden wird. $m' = m_{sum}$ ist die Anzahl der „Ja“-Stimmen.

Die sichere Übertragung von r nach T kann mit der ElGamal Kryptosystem-Verschlüsselung stattfinden, da die öffentlichen Parameter schon für das ElGamal Commitment generiert wurden und dazu wieder verwendet werden können. Durch diese unkonventionelle Nutzung eines Commitment-Verfahrens können sich die Wähler selbst von dem Wahlergebnis überzeugen und sicher sein, dass die Stimmen geheim und bindend sind. Der klare Nachteil hierbei liegt in der Abhängigkeit von der vertrauten Person T, die benötigt wird um r_{sum} zu berechnen. Das Senden des Zufallsmaterials jedes Wählers an T ermöglicht es ihr auf Verlangen jedes Commitment öffnen zu können.

7 Zusammenfassung und Ausblick

Rückblickend wurde in dieser Arbeit der Einstieg in das Thema Commitment-Verfahren geleistet. Commitment-Verfahren werden von Sendern benutzt, um sich zu einem Wert zu verpflichten und diesen später enthüllen zu können. Dazu wird diese Verpflichtung verschlüsselt, um sie so den Empfängern unkenntlich zu machen. Die Sicherheit der Verschlüsselung basiert auf mathematischen Problemen, zu denen noch keine effiziente Lösung bekannt ist. Anders als bei beispielsweise Kryptosystemen, die eine Nachricht vor Dritten geheim halten sollen, liegt der primäre Fokus bei Commitment-Verfahren darin, den eigentlichen Wert m eines Commitments C vor dem Empfänger geheim zu halten. Bis der Sender bereit ist diesen mitzuteilen. Aus diesem Grund existieren keine Entschlüsselungsfunktionen, welche der Empfänger anwenden kann. Als Beweis der Verpflichtung genügt es, dem Empfänger die Daten zur Erstellung eines Commitments C zu geben. Dadurch hat er selbst die Möglichkeit ein Commitment zu erzeugen und dies mit seinem Erhaltenem zu vergleichen.

Zur Sicherheit von erzeugten Commitments der Commitment-Verfahren wurden deren Sicherheitseigenschaften Hiding und Binding anhand eines generischen Commitment-Verfahrens in Kapitel 3 erläutert. Hiding beschützt den Sender des Commitments vor dem vorzeitigen Aufdecken des Wertes und Binding beschützt den Empfänger vor Betrug des Senders, damit dieser auch wirklich an seinen originalen Wert gebunden ist. Diese beiden Eigenschaften lassen sich in den zwei Varianten Unconditional und Computational aufteilen. Ist die Eigenschaft Unconditional, so kann sie auch mit unbegrenzter Rechenkraft nicht gebrochen werden. Wenn sie Computational ist, hat ein Angriff auf die Eigenschaft ohne unbegrenzte Rechenkraft nur eine vernachlässigbare Chance auf Erfolg. Ein Commitment-Verfahren, welches beide Sicherheitseigenschaften in der Unconditional-Variante erfüllt, existiert nicht, wie in Beweis 1 gezeigt wurde.

Das Public-Key-Kryptosystem RSA wurde in Kapitel 4 vorgestellt und zu einem Commitment-Verfahren adaptiert. Aufgrund von Satz 2 muss zusätzlich zur Erfüllung der Hiding-Eigenschaft bei dem Verpflichten eine Zufallszahl an die zu verpflichtende Zahl beigefügt werden. Die Nutzung des adaptierten RSA Commitment-Verfahrens wurde exemplarisch vorgeführt. Daraufhin wurden kryptographische Hashfunktionen als potentielle Commitment-Verfahren betrachtet, weil deren Konzept der Commit-Phase von Commitment-Verfahren gleicht. Als Ergebnis stellte sich heraus, dass diese keine der Sicherheitseigenschaften in der Unconditional-Variante besitzen. Ein Vorteil dieser Nutzung wäre jedoch, dass die Nachricht beliebig gewählt werden kann und nicht als Voraussetzung vorerst in Zahlen bestimmter Intervalle umgewandelt werden muss, wie bei anderen Verfahren.

Die Commitment-Verfahren basierend auf Pedersen und ElGamal wurden jeweils in Kapitel 5 und 6 vorgestellt und deren Ähnlichkeit zueinander gezeigt. Trotz der Ähnlichkeit erfüllen sie jeweils die gegenseitige Sicherheitseigenschaft in der Unconditional-

Variante. Die Nutzung der beiden Verfahren wurde an einem Beispiel illustriert und jeweils ein möglicher Anwendungsbereich gezeigt.

In Tabelle 7 sind alle in dieser Arbeit untersuchten Commitment-Verfahren und deren Eigenschaften zusammengetragen.

	Sicherheitseigenschaften		
Name	Binding	Hiding	Besonderheit
RSA	Unconditional	Computational	-
krypt. Hashf.	Computational	Computational	freie Wahl für m, r
Pedersen	Computational	Unconditional	additiv homomorph
ElGamal	Unconditional	Computational	additiv homomorph

Tabelle 7.1: Commitmentverfahren Übersicht

Als Fazit lässt sich sagen, dass das verwendete Commitment-Verfahren jeweils anwendungsbezogen gewählt werden sollte, mit Bedacht in welcher Variation die Sicherheitseigenschaften benötigt werden. Auch die adaptierten Verfahren können benutzt werden. Wenn in der Infrastruktur eines Systems bereits das RSA-Verfahren verwendet wird, kann das RSA Commitment-Verfahren ohne große Veränderung des Systems hinzugefügt werden. Commitment-Verfahren basierend auf kryptographischen Hashfunktionen können ebenso Einsatz finden, beispielsweise bei Projekten, in denen die inhärente Sicherheit von Hashfunktionen ausreicht.

Ausblick In der theoretischen Informatik können Commitment-Verfahren im Bezug zu dem Thema Zero-Knowledge benutzt werden. Zero-Knowledge Beweise erlauben es den Besitz oder Kenntnis einer Information zu beweisen. Das besondere hierbei ist, dass zu dem Beweis keine weiteren Informationen benötigt werden um das Gegenüber zu überzeugen. Kapitel 4 des Lehrbuchs von Goldreich beschäftigt sich mit diesem Thema und zeigt, dass durch Commitment-Verfahren die Existenz von Zero-Knowledge Beweisen für jede Sprache in NP gezeigt werden kann [Gol01].

Das Konzept eines fuzzy Commitment-Verfahren wurde von Juels und Wattenberg auf einer Konferenz vorgestellt. Dies ist ein Commitment-Verfahren mit der Eigenschaft Commitments nicht nur mit der genauen ursprünglichen Nachricht m zu beweisen, sondern auch mit Nachrichten m', die ähnlich genug zu m sind. Der Zweck hierzu ist bei biometrischen Systemen gedacht, beispielsweise können Retina-Scans von demselben Auge leicht verschieden sein aufgrund von kleinen zufälligen Signalstörungen [JW99].

Non-malleability ist ein weiteres Konzept in der Kryptographie, welches auf Commitment-Verfahren übertragen werden kann. Ein non-malleable Commitment-Verfahren bedeutet, dass es nicht zulässig ist aus einem bekannten Commitment C mit einem unbekannten Wert m ein valides Commitment C' mit einem Wert $m + 1$ zu generieren. Marc und Roger Fischlin stellten auf einer Konferenz ein Verfahren mit diesem Konzept vor und verglichen es mit anderen existierenden non-malleable Commitment-Verfahren [FF00].

Trapdoor Commitment-Verfahren sind eine erweiterte Form von Commitment-Verfahren. Sie erzeugen Commitments mit der zusätzlichen Eigenschaft einer geheimen Hintertür. Kennt man diese Hintertür lässt sich der bereits verpflichtete Wert nach der Commit-Phase noch ändern. Informationen zu und die ermöglichten Applikationen von Trapdoor Commitment-Verfahren werden in einer Arbeit von Marc Fischlin gezeigt [Fis01].

Literaturverzeichnis

[Ada] AdamISZ. Github - adamisz/confidentialtransactionsdoc: A detailed description of how bitcoin elements alpha confidential transactions works. `https://github.com/AdamISZ/ConfidentialTransactionsDoc`. (Abrufdatum: 04.10.2020).

[BD06] Elaine Barker and Quynh Dang. NIST Special Publication 800-57 Part 3 Revision 1: Recommendation for key management: Application-specific key management guidance. 2006. URL: `http://dx.doi.org/10.6028/NIST.SP.800-57pt3r1`, `doi:10.6028/NIST.SP.800-57pt3r1`.

[Blu81] Manuel Blum. Coin flipping by telephone. In Allen Gersho, editor, *Advances in Cryptology: A Report on CRYPTO 81, CRYPTO 81, IEEE Workshop on Communications Security, Santa Barbara, California, USA, August 24-26, 1981*, pages 11–15. U. C. Santa Barbara, Dept. of Elec. and Computer Eng., ECE Report No 82-04, 1981.

[BR94] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In Alfredo De Santis, editor, *Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer, 1994. `doi:10.1007/BFb0053428`.

[Buc04] Johannes A. Buchmann. *Einführung in die Kryptographie, 3. Auflage*. Springer, 2004.

[BW13] David Bernhard and Bogdan Warinschi. Cryptographic voting - A gentle introduction. In Alessandro Aldini, Javier López, and Fabio Martinelli, editors, *Foundations of Security Analysis and Design VII - FOSAD 2012/2013 Tutorial Lectures*, volume 8604 of *Lecture Notes in Computer Science*, pages 167–211. Springer, 2013. `doi:10.1007/978-3-319-10082-1\_7`.

[DN08] Ivan Damgård and Jesper Buus Nielsen. Commitment schemes and zero-knowledge protocols. 2008. URL: `https://users-cs.au.dk/ivan/ComZK08.pdf`.

[Dud] "Commitment" auf Duden online. `https://www.duden.de/node/29136/revision/29165`. (Abrufdatum: 04.10.2020).

[FF00] Marc Fischlin and Roger Fischlin. Efficient non-malleable commitment schemes. In Mihir Bellare, editor, *Advances in Cryptology - CRYPTO 2000, 20th*

Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings, volume 1880 of *Lecture Notes in Computer Science*, pages 413–431. Springer, 2000. doi:10.1007/3-540-44598-6_26.

[Fis01] Marc Fischlin. *Trapdoor commitment schemes and their applications*. PhD thesis, Goethe University Frankfurt, Frankfurt am Main, Germany, 2001. URL: http://zaurak.tm.informatik.uni-frankfurt.de/diss/data/src/00000229/00000229.pdf.gz.

[Gam85] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inf. Theory*, 31(4):469–472, 1985. doi:10.1109/TIT.1985.1057074.

[Gol01] Oded Goldreich. *The Foundations of Cryptography - Volume 1: Basic Techniques*. Cambridge University Press, 2001. URL: http://www.wisdom.weizmann.ac.il/%7Eoded/foc-vol1.html, doi:10.1017/CBO9780511546891.

[JW99] Ari Juels and Martin Wattenberg. A fuzzy commitment scheme. In Juzar Motiwalla and Gene Tsudik, editors, *CCS '99, Proceedings of the 6th ACM Conference on Computer and Communications Security, Singapore, November 1-4, 1999*, pages 28–36. ACM, 1999. doi:10.1145/319709.319714.

[KL07] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Chapman and Hall/CRC Press, 2007. URL: http://www.cs.umd.edu/%7Ejkatz/imc.html.

[Ped91] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer, 1991. doi:10.1007/3-540-46766-1_9.

[Rot08] Jörg Rothe. *Komplexitätstheorie und Kryptologie. Eine Einführung in Kryptokomplexität*. eXamen.press. Springer, 2008. doi:10.1007/978-3-540-79745-6.

Erklärung der Selbstständigkeit

Hiermit versichere ich, die vorliegende Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet zu haben. Die Arbeit hat in gleicher oder ähnlicher Form noch keinem anderen Prüfungsamt vorgelegen.

Hannover, den 12. Oktober 2020

_____

Rico Seebonn