Hochschule RheinMain
Fachbereich Design Informatik Medien
Studiengang Angewandte Informatik

Abschlussarbeit

zur Erlangung des akademischen Grades

Bachelor of Science (B.Sc.)

Homomorphe Verschlüsselung

Vorgelegt von	Viola Campos
am	17. August 2014
Referent	Prof. Dr. Steffen Reith
Korreferent	Prof. Dr. Bernhard Geib

Erklärung gemäß ABPO

Ich erkläre hiermit,

- dass ich die vorliegende Abschlussarbeit selbstständig angefertigt,
- keine anderen als die angegebenen Quellen benutzt,
- die wörtlich oder dem Inhalt nach aus fremden Arbeiten entnommenen Stellen, bildlichen Darstellungen und dergleichen als solche genau kenntlich gemacht und
- keine unerlaubte fremde Hilfe in Anspruch genommen habe.

Wiesbaden, 17. August 2014 _____

Viola Campos

Erklärung zur Verwendung der Bachelor Thesis

Hiermit erkläre ich mein Einverständnis mit den im folgenden aufgeführten Verbreitungsformen dieser Abschlussarbeit:

Verbreitungsform	**Ja**	**Nein**
Einstellung der Arbeit in die Hochschulbibliothek mit Datenträger		×
Einstellung der Arbeit in die Hochschulbibliothek ohne Datenträger	×	
Veröffentlichung des Titels der Arbeit im Internet	×	
Veröffentlichung der Arbeit im Internet		×

Wiesbaden, 17. August 2014 _____

Viola Campos

Zusammenfassung

Voll homomorphe Verschlüsselung (Fully Homomorphic Encryption, FHE) ermöglicht beliebig komplexe Berechnungen auf verschlüsselten Daten. Das Konzept von Rechnen unter Verschlüsselung wurde erstmals 1978 von Rivest, Adleman und Dertouzos formuliert [RAD78] und galt für die folgenden 30 Jahre als «heiliger Gral»der Kryptographie. Ein solches Verfahren wurde als extrem nützlich aber nicht realisierbar eingeschätzt. Erst 2009 konstruierte Craig Gentry das erste funktionierende voll homomorphe Verschlüsselungssystem [Gen09], das allerdings aufgrund seiner Komplexität praktisch nicht einsetzbar war. Die Theorie hinter Gentrys Ansatz inspirierte in den Folgejahren eine Reihe weiterer Arbeiten zum Thema, so dass einfachere FHE Systeme basierend auf neuen mathematischen Techniken sowie unterschiedliche Anwendungen entstanden.

Diese Arbeit teilt sich in einen theoretischen und einen praktischen Teil. Der erste Teil gibt einen Überblick über den aktuellen Stand der Forschung im Bereich der homomorphen Verschlüsselung und zeigt Motivationen für deren Einsatz sowie aktuelle Probleme auf. Weiterhin werden die zugrunde liegenden Ideen und Konzepte vorgestellt.

Den zweiten Teil der Arbeit bildet die Analyse und prototypische Implementierung eines aktuellen homomorphen Verschlüsselungssystems. Gewählt wurde das Approximate Eigenvector Schema von Gentry, Sahai und Waters von 2013 [GSW13]. In der entstandenen Implementierung wurden das System ergänzt durch Optimierungen von Brakerski und Vaikuntanathan [BV14], die die homomorphe Auswertung komplizierterer Funktionen ermöglichten. Die Leistung der Implementierung wird analysiert und in einem abschließenden Fazit werden weitere Optimierungsmöglichkeiten sowie ein Ausblick auf die weitere Entwicklung dargestellt.

Inhaltsverzeichnis

Kapitel 1

Einführung

Klassische Kryptographie bezeichnet die Wissenschaft zur Erforschung und Realisierung von Verfahren zur Verschlüsselung von Informationen. In den letzten 30 Jahren wurde der Begriff immer weiter gefasst, unter moderner Kryptographie versteht man heute allgemein Techniken, die die Sicherheit digitaler Informationen, Transaktionen und verteilter Rechensysteme schützen [KL07].

Kryptographische Verfahren zur Verschlüsselung von Informationen ermöglichen es zwei Parteien, Nachrichten über einen öffentlichen Kanal auszutauschen, ohne dass Dritte unberechtigt auf die Daten zugreifen können. Zur Veranschaulichung heißen die beiden Parteien in dieser Arbeit Alice und Bob. Möchte Alice eine geheime Nachricht an Bob schicken, verschlüsselt sie die Daten mit einem geheimen Schlüssel. Selbst wenn die neugierige Eve die verschlüsselten Daten während der Übertragung abhört, kann sie daraus keine Informationen über die eigentliche Nachricht ableiten. Erst Bob, der den passenden Schlüssel für die Entschlüsselung der Daten besitzt, kann die Klartexte wieder herstellen.

Verfahren zur sicheren Nachrichtenübermittlung werden mittlerweile in vielen Kommunikationsprotokollen als Standard eingesetzt. Beispielsweise werden im Internet Anfragen an große Suchmaschinen standardmäßig SSL[1] -verschlüsselt übertragen. Damit sind die Daten auf dem Transportweg vor dem unberechtigten Zugriff durch Dritte geschützt, der Empfänger entschlüsselt die Daten, um die Suche durchzuführen und überträgt die Suchergebnisse ebenfalls verschlüsselt (siehe Abbildung 1.1). Der Suchmaschinenbetreiber als Empfänger der Nachricht kennt notwendigerweise den Inhalt der Anfrage, was kein Problem darstellt, so lange die Nutzer ihn als vertrauenswürdig wahrnehmen. Homomorphe Verschlüsselung bietet eine Lösung für Szenarien, in denen das nicht der Fall ist.

[1] Secure Sockets Layer, Verschlüsselungsprotokoll zur sicheren Datenübertragung im Internet

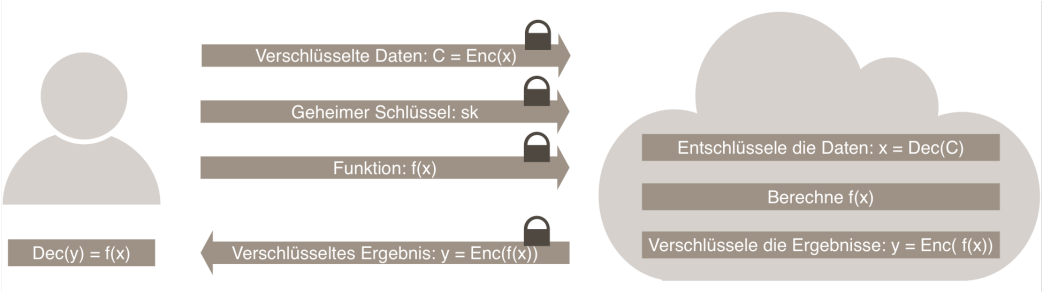

Abbildung 1.1: Klassische Verschlüsselung

In einer homomorph verschlüsselten Variante des Beispiels (siehe Abbildung 1.2) sendet Alice eine verschlüsselte Suchanfrage an Bobs Suchmaschine, Bob führt diese aus, *ohne* diese zu entschlüsseln, also ohne etwas über den Inhalt der Suche zu erfahren und gibt das ebenfalls verschlüsselte Ergebnis des Suchalgorithmus an Alice zurück. Alice, die als Einzige den geheimen Schlüssel für die Entschlüsselung der Daten kennt, kann dieses zum eigentlichen Suchergebnis entschlüsseln.

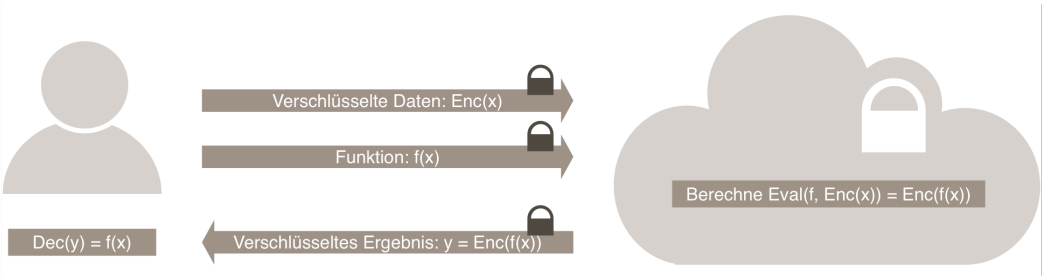

Abbildung 1.2: Homomorphe Verschlüsselung

Homomorphe Verschlüsselung beschreibt demnach eine Form der Verschlüsselung, welche es ermöglicht, Berechnungen auf verschlüsselten Daten auszuführen und so ein ebenfalls verschlüsseltes Ergebnis zu erzeugen. Dessen Entschlüsselung entspricht dem Ergebnis der Berechnungen auf den ursprünglichen Klartexten.

1.1 Motivation

Kryptographische Systeme, die Rechenoperationen auf verschlüsselten Daten ermöglichen, sind für viele Anwendungen interessant. Das vermutlich wichtigste potentielle Einsatzgebiet homomorpher Verschlüsselung stellt aktuell das Cloud Computing dar [Vai11]. Während vom Anwender genutzte (mobile) Endgeräte oft über vergleichsweise geringe Rechenleistung verfügen, besteht in zunehmend service-orientierten IT-Infrastrukturen gleichzeitig Zugang zu leistungsstarken Servern in der Cloud, es wäre also wünschenswert, rechenintensive Operationen aus-

zulagern.

Eine wichtige Frage ist in diesem Zusammenhang wie vertrauliche Daten der Nutzer eines Cloud Services vor dem unberechtigten Zugriff durch den Dienstanbieter oder andere Nutzer des Dienstes geschützt werden können. Während der Nutzer eines reinen Cloud Storage Dienstes die Möglichkeit hat, seine Daten verschlüsselt abzulegen, existiert diese Möglichkeit für Nutzer eines Cloud Computing Dienstes aktuell noch nicht. Homomorphe Verschlüsselung könnte diese Lücke schliessen. Ein Cloud-Anbieter könnte beliebige Berechnungen wie die Suche nach bestimmten Datensätzen oder statistische Analysen auf den verschlüsselten Daten seiner Nutzer vornehmen und dem Besitzer der Daten die Ergebnisse in ebenfalls verschlüsselter Form zur Verfügung stellen. Dabei sind neben den Daten auch die eingesetzten Algorithmen als geistiges Eigentum des Dienstanbieters vor unberechtigtem Zugriff geschützt.

Konkrete Anwendungsmöglichkeiten eines solchen sicheren Cloud Computing Services sind zum Beispiel die Auswertung medizinischer Daten, quantitative Analysen von Börsendaten oder biometrische Analysen von Aufnahmen öffentlicher Überwachungskameras (siehe auch Abbildung 1.3).

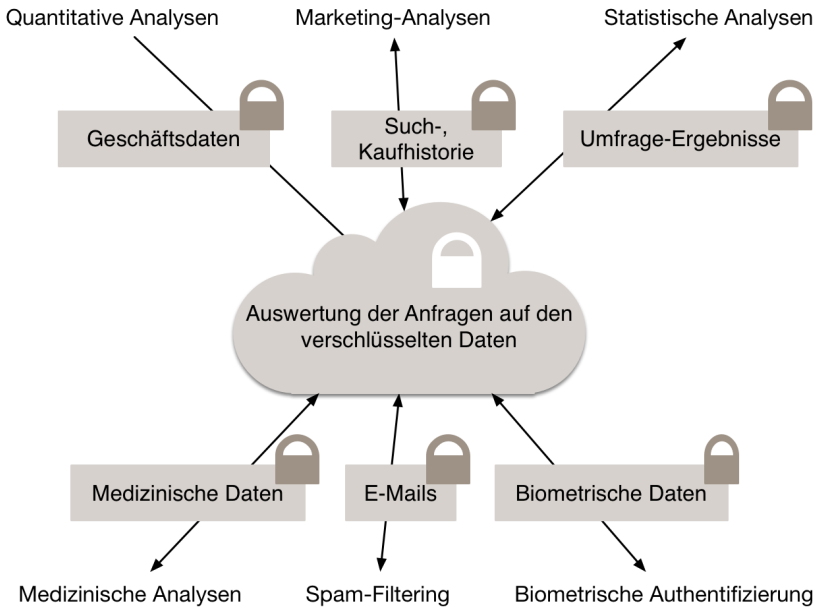

Abbildung 1.3: Anwendungsmöglichkeiten für sichere Cloud Services

1.2 Entwicklung homomorpher Verschlüsselung

Die Idee eines kryptographischen Systems, welches nicht-triviale Berechnungen auf verschlüsselten Daten ermöglicht, ohne diese vorher zu entschlüsseln, wurde erstmals 1978, ein Jahr nach der Entwicklung von RSA von Rivest, Adleman und Dertouzos formuliert [RAD78].

Teilweise homomorphe Verschlüsselung

In den folgenden 30 Jahren entstanden einige *teilweise homomorphe Verschlüsselungssysteme*, die entweder homomorphe Eigenschaften bezüglich der Addition oder der Multiplikation verschlüsselter Daten aufweisen. Mit Hilfe dieser Systeme lassen sich Anwendungen realisieren, deren homomorph auszuwertende Funktionen sich ausschließlich aus aufeinanderfolgenden Additionen oder Multiplikationen aufbauen lassen, die aber keine Kombination aus beiden Operationen benötigen. Einen Überblick über teilweise homomorphe Systeme bietet Tabelle 1.1.

Kryptosystem	hom. Eigenschaft	Jahr	praktische Anwendung
Unpadded RSA	Multiplikation	1977	
Goldwasser-Micali	binäres XOR	1984	PIR, Auktionssysteme mit verdeckten Geboten [PBD05], biometrische Authentifizierung [BCI+07]
El Gamal	Multiplikation	1984	
Benaloh	Addition	1994	PIR, E-Voting, E-Cash [Vai11]
Okamoto-Uchiyama	Addition	1998	PIR, E-Voting, E-Cash [Vai11]
Naccache-Stern	Addition	1998	PIR, E-Voting, E-Cash [Vai11]
Paillier	Addition	1999	PIR, E-Voting, E-Cash [Vai11]
Dåmgard-Jurik	Addition	2001	PIR, E-Voting, E-Cash [Vai11]

Tabelle 1.1: Teilweise homomorphe Verschlüsselungssysteme

Additive homomorphe Systeme werden aktuell bereits in verschiedenen Anwendungen eingesetzt [Vai11]. Beispielsweise kommt das von Cohen und Fischer entwickelte Kryptosystem [CF85] in verschiedenen kommerziell genutzten E-Voting Systemen zum Einsatz. Die additiv homomorphe Eigenschaft des Systems erlaubt es, Wählerstimmen zu zählen, ohne vertrauliche Informationen über die einzelnen Wähler offenzulegen.

Ein weiteres wichtiges Einsatzgebiet für additiv homomorphe Verschlüsselung sind

Protokolle zum Private Information Retrieval (PIR). Diese ermöglichen es einem Nutzer, einen bestimmten, durch einen Index i identifizierten Eintrag aus einer Datenbank auf einem entfernten Server anzufordern, ohne dem Eigentümer der Datenbank Informationen über den gewünschten Index i offenzulegen. Kushilevitz und Ostrovsky [KO97] entwickelten ein auf additiv homomorpher Verschlüsselung basierendes, sehr effektives PIR Protokoll.

Voll homomorphe Verschlüsselung

Das erste System, dessen homomorphe Eigenschaften über rein additive oder multiplikative Homomorphie hinausgingen, wurde 2005 von Boneh, Goh und Nissim konstruiert [BGN05]. Es erlaubt die homomorphe Auswertung einer Folge von beliebig vielen Additionen sowie einer Multiplikation. Die begrenzte Anzahl möglicher aufeinanderfolgender Operation ergibt sich aus der probabilistischen Verschlüsselung homomorpher Verfahren. Um gleiche Klartexte auf unterschiedliche Geheimtexte abzubilden, sind die verschlüsselten Daten mit kleinen numerischen Fehlern behaftet, also leichten Abweichungen von ihren idealen Werten. Während der Entschlüsselung findet eine versteckte Fehlerkorrektur statt, die allerdings nur funktioniert, solange die Abweichungen klein genug sind. Jede arithmetische Operation auf den Geheimtexten verstärkt dieses Rauschen. Näherungsweise lässt sich sagen, dass jede homomorphe Addition den Fehler verdoppelt, jede Multiplikation quadriert ihn. Ab einem gewissen Betrag überdeckt das Rauschen die eigentliche Nachricht und diese lässt sich nicht mehr fehlerfrei entschlüsseln.

Das erste System, das diese Problem theoretisch lösen konnte und beliebig komplexe Berechnungen auf verschlüsselten Daten ermöglichte, wurde 2009 von Craig Gentry konstruiert [Gen09]. Neu an seinem Verfahren war der sogenannte *Bootstrapping*-Vorgang, der Geheimtexte, welche mit grossen Fehlern behaftet waren, in ein Format mit reduziertem Fehler überführen konnte.

In den folgenden Jahren entstanden weitere Konstruktionen voll homomorpher Verschlüsselungssysteme, die im wesentlichen Gentrys Vorgehen auf anderen kryptographische Probleme übertrugen oder bestehende Systeme weiter optimierten. Smart und Vercauteren [SV09] sowie Gentry und Halevi [GH10] entwickelten optimierte Varianten von Gentrys auf idealen Gittern (siehe Kapitel 2.5) beruhendem Schema mit reduzierter Komplexität. Van Dijk, Gentry, Halevi und Vaikuntanathan konstruierten ein FHE Schema, dessen Sicherheit auf einem Problem der Ganzzahlarithmetik basiert und welches nur elementare arithmetische Operationen auf ganzen Zahlen nutzt [vDGHV10]. Ein gemeinsames Problem dieser Systeme war, dass die zugrunde liegenden Probleme in idealen Gittern, welche die Sicherheit der Systeme garantierten, noch wenig untersucht sind, es war also ein Ziel, FHE

Systeme zu entwickeln deren Sicherheit auf besser untersuchten kryptographischen Standardannahmen basiert. Erreicht wurde dies durch Systeme, die auf dem *Learning with Error* Problem (LWE) [BV11a, Bra12, GSW13] oder dem Ring-LWE Problem aufbauen [BV11b, BGV12]. Am vielversprechendsten für die weitere Entwicklung erscheinen aktuell Ring-LWE Systeme sowie eine auf dem NTRU Verschlüsselungssystem [2] basierende Variante von Lopéz-Alt, Tromer und Vaikuntanathan [LATV12]. Auch wenn diese Systeme bereits deutlich effizienter arbeiten als die ersten FHE Konstruktionen von 2009 sind sie aktuell noch zu komplex für den praktischen Einsatz.

1.3 Gliederung der Thesis

Die Arbeit ist folgendermaßen aufgebaut. Kapitel 2 erläutert allgemeine, zum Verständnis der Arbeit notwendige Grundlagen, in Kapitel 3 werden homomorphe Verschlüsselung und die Anforderungen an ein solches System formal definiert. Kapitel 4 stellt als konkretes Beispiel eines aktuellen homomorphen Verschlüsselungssystems das FHE Schema von Gentry, Sahai und Waters [GSW13] vor. Das Schema wurde 2013 veröffentlicht und basiert wie einige andere aktuelle FHE Systeme auf dem Learning with Errors Problem. Neu an dem Schema ist die Form der Geheimtexte, welche die Komplexität homomorpher Berechnungen reduzieren soll. Kapitel 5 stellt die Implementierung des Verfahrens vor, während in Kapitel 6 die Ergebnisse der Analyse zusammengefasst werden.

[2] NTRU: N-Th Degree Truncated Polynomial Ring. Gitterbasiertes asymmetrisches Verschlüsselungssystem

Kapitel 2

Grundlagen

Das folgende Kapitel stellt im ersten Teil für das Verständnis der in dieser Arbeit vorgestellten Verfahren und Konzepte wichtige mathematische Grundlagen zusammen, der zweite Teil bietet dann einen Überblick über kryptographische Grundlagen.

2.1 Notationen

Diese Arbeit verwendet Notationen in Anlehnung an [TT08, BV14] und [GSW13]. Skalare werden als normale, Vektoren als fettgedruckte Kleinbuchstaben notiert und Matrizen werden durch fettgedruckte Großbuchstaben dargestellt. Die folgende Liste dient als Überblick über die verwendeten mathematische Symbole und ihre Definitionen.

Notation	Definition
$\log x$	Logarithmus zur Basis 2: $\log_2 x$
$\mathbb{N}$	Menge der natürlichen Zahlen ohne Null: $\{1, 2, \dots\}$
$\mathbb{Z}$	Menge der ganzen Zahlen: $\{\dots, -2, -1, 0, 1, 2, \dots\}$
$\mathbb{Z}_q$	Ring der ganzen Zahlen modulo q für eine positive ganze Zahl q
$\mathbb{Z}^m$	Menge der m-dimensionalen Vektoren über $\mathbb{Z}$
$\mathbb{Z}^{m \times n}$	Menge der $m \times n$-Matrizen über $\mathbb{Z}$
$\mathbf{I}_m$	Einheitsmatrix der Dimension m, $\mathbf{I}_m \in \mathbb{Z}^{m \times m}$
$\|\mathbf{v}\|_i$	l_i-Norm eines Vektors $\mathbf{v}$
$\mathbf{v}[i]$	Das i-te Element eines Vektors $\mathbf{v}$
$\langle \mathbf{v}, \mathbf{u} \rangle$	Skalarprodukt der Vektoren $\mathbf{u}$ und $\mathbf{v}$
$\lfloor x \rfloor$	Abrunden von x auf nächste natürliche Zahl: $max\{z \in \mathbb{Z} \mid z \leq x\}$
$\lceil x \rceil$	Aufrunden von x auf nächste natürliche Zahl: $min\{z \in \mathbb{Z} \mid z \geq x\}$
$\lfloor x \rceil$	Runden: $\lfloor x \rfloor$ für $x - \lfloor x \rfloor < \frac{1}{2}$, sonst $\lceil x \rceil$

$\|x\|$	Betrag von x: x für $x \geq 0$, $-x$ für $x < 0$
$[x]_q$	x modulo q
$x \sim \mathcal{D}$	Ist $\mathcal{D}$ eine Wahrscheinlichkeitsverteilung auf einer Grundmenge X, so bezeichnet $x \sim \mathcal{D}$ ein zufällig $\mathcal{D}$-verteiltes $x \in X$.
$X \leftarrow \mathsf{A}(Y)$	Bezeichnet A einen probabilistischen Algorithmus und Y eine Eingabe für A, dann bezeichnet $X \leftarrow \mathsf{A}(Y)$ eine mögliche Ausgabe des Algorithmus.
$O(f(n))$	Asymptotische obere Schranke: Die Menge aller Funktionen $g(n)$, für die es zwei Konstanten n_0 und c gibt, so dass für alle $n \geq n_0$ gilt: $g(n) \leq c \cdot f(n)$
$\tilde{O}(f(n))$	Logarithmus-bereinigte asymptotische obere Schranke: Die Menge aller Funktionen $g(n)$, für die es drei Konstanten n_0, c und k gibt, so dass für alle $n \geq n_0$ gilt: $g(n) \leq c \cdot f(n) \cdot \log^k f(n)$

2.2 Algebraische Grundlagen

Asymmetrische kryptographische Verfahren basieren in der Regel auf Operationen in diskreten mathematischen Strukturen, beispielsweise nutzt das in dieser Arbeit vorgestellte Verfahren den Ring der ganzen Zahlen modulo einer positiven ganzen Zahl q. Folgende Definitionen zeigen die Eigenschaften der für diese Arbeit relevanten algebraischen Strukturen:

Definition 2.1 (Gruppe [TT08])**.** *Sei G eine Menge mit einer Verknüpfung, die je zwei Elementen $a, b \in G$ ein Element $a \circ b \in G$ zuordnet. Dann wird $(G, \circ)$ eine* Gruppe *genannt, wenn folgendes gilt:*

1. *Es gilt $(a \circ b) \circ c = a \circ (b \circ c)$ für alle $a, b, c \in G$. (*Assoziativgesetz*)*
2. *Es gibt ein* neutrales Element *$n \in G$, das $n \circ a = a \circ n = a$ für alle $a \in G$ erfüllt.*
3. *Zu jedem $a \in G$ gibt es ein inverses Element $i(a) \in G$, das $a \circ i(a) = i(a) \circ a = n$ erfüllt.*

Gilt zusätzlich

4. *$a \circ b = b \circ a$ für alle $a, b \in G$,* (Kommutativgesetz)

so spricht man von einer kommutativen *oder* abelschen Gruppe*.*

Definition 2.2 (Ring [TT08])**.** *Eine Menge R mit zwei Verknüpfungen $+$ und $\cdot$, geschrieben $(R, +, \cdot)$ heisst* Ring*, wenn folgendes gilt:*

1. *$(R, +)$ ist eine kommutative Gruppe mit neutralem Element 0.*
2. *Für alle $a, b, c \in R$ gilt: $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ (*Assoziativgesetz*)*
3. *Für alle $a, b, c \in R$ gilt: $a \cdot b + a \cdot c = a \cdot (b + c)$ (*Distributivgesetz*).*

Gilt zusätzlich

4. *das* Kommutativgesetz $a \cdot b = b \cdot a$ *für alle* $a, b \in R$*, so spricht man von einem kommutativen Ring, und wenn darüber hinaus*
5. *ein* neutrales Element 1 für die Multiplikation *existiert, also* $a \cdot 1 = 1 \cdot a = a$ *für alle* $a \in R$*,*

so spricht man von einem kommutativen Ring mit Eins*.*

Definition 2.3 (Körper [TT08])**.** *Eine Menge* $\mathbb{K}$ *mit zwei Verknüpfungen* $+$ *und* $\cdot$*, geschrieben* $(\mathbb{K}, +, \cdot)$ *heisst* Körper*, wenn folgendes gilt:*

1. $(\mathbb{K}, +)$ *ist eine kommutative Gruppe mit neutralem Element 0.*
2. $(\mathbb{K} \setminus \{0\}, \cdot)$ *ist eine kommutative Gruppe mit neutralem Element 1.*
3. *Für alle* $a, b, c \in \mathbb{K}$ *gilt:* $a \cdot b + a \cdot c = a \cdot (b + c)$ *(*Distributivgesetz*).*

Bei $(\mathbb{Z}_q, +, \cdot)$ handelt es sich um einen kommutativen Ring mit Eins. Gilt zusätzlich die Einschränkung, dass es sich bei q um eine Primzahl handelt, ist $(\mathbb{Z}_q, +, \cdot)$ ein *Körper* [TT08].

2.2.1 Homomorphismus

Als Homomorphismus bezeichnet man in der Algebra eine strukturerhaltende Abbildung zwischen zwei algebraischen Strukturen vom selben Typ. Ein Homomorphismus bildet die Elemente aus der einen Menge so in die andere Menge ab, dass sich ihre Bilder dort hinsichtlich der Struktur genauso verhalten, wie deren Urbilder in der Ausgangsmenge. Betrachtet man beispielsweise eine Abbildung zwischen zwei Ringen $(R, +, \cdot)$ und $(S, \oplus, \otimes)$, dann wird die Abbildung $f : R \to S$ Ringhomomorphismus genannt, wenn für jedes Element a, b von R gilt [TT08]:

$$f(a + b) = f(a) \oplus f(b) \text{ und}$$
$$f(a \cdot b) = f(a) \otimes f(b)$$

Es ist also egal, ob man zwei Elemente verknüpft und das Ergebnis abbildet, oder erst die Abbildungen der Elemente bildet und dann die Bilder verknüpft.

2.3 Vektoren und Matrizen

Das vorgestellte Verfahren nutzt verschiedene Transformationen auf Vektoren und Matrizen, deren Grundlagen im folgenden Abschnitt zusammengestellt werden.

Ein n-Tupel mit Einträgen aus einem Körper $\mathbb{K}$ wird *Vektor* über $\mathbb{K}$ genannt. Die Menge der Vektoren mit jeweils n Einträgen aus $\mathbb{K}$ bilden einen n-dimensionalen

Vektorraum V über $\mathbb{K}$. Die Rechenoperationen in einem Vektorraum sind folgendermassen definiert:

Definition 2.4 (Vektorraum [TT08])**.** *Sei* $\mathbb{K}$ *ein beliebiger Körper und* V *eine Menge mit zwei Verknüpfungen:*

1. Vektoraddition: *Je zwei Elementen* $\mathbf{a}, \mathbf{b} \in V$ *wird ein Element* $\mathbf{a} + \mathbf{b} \in V$ *zugeordnet, so dass* $(V, +)$ *eine kommutative Gruppe wird, das heisst*

$$\begin{aligned} \mathbf{a} + \mathbf{b} &= \mathbf{b} + \mathbf{a} && \textit{Kommutativität} \\ \mathbf{a} + (\mathbf{b} + \mathbf{c}) &= (\mathbf{a} + \mathbf{b}) + \mathbf{c} && \textit{Assoziativität} \\ \mathbf{a} + \mathbf{0} &= \mathbf{a} && \textit{Existenz des neutralen Elements} \\ \mathbf{a} + (-\mathbf{a}) &= \mathbf{0} && \textit{Existenz des inversen Elements} \end{aligned}$$

   für alle $\mathbf{a}, \mathbf{b}, \mathbf{c} \in V$*. Das neutrale Element* $\mathbf{0}$ *ist der* Nullvektor *und das zu* $\mathbf{a}$ *inverse Element ist* $-\mathbf{a}$*.*
2. Multiplikation mit einem Skalar: *Je einem* $\mathbf{a} \in V$ *und einem* $k \in \mathbb{K}$ *wird ein Element* $k\mathbf{a} \in V$ *zugeordnet, so dass*

$$\begin{aligned} k(h\mathbf{a}) &= (kh)\mathbf{a} && \\ 1\mathbf{a} &= \mathbf{a} && \\ k(\mathbf{a} + \mathbf{b}) &= k\mathbf{a} + k\mathbf{b} && \textit{Distributivgesetze} \\ (k + h)\mathbf{a} &= k\mathbf{a} + h\mathbf{a} && \end{aligned}$$

   für alle $\mathbf{a}, \mathbf{b} \in V$ *und* $h, k \in \mathbb{K}$*.*

Dann wird V *ein* Vektorraum *(über dem Körper* $\mathbb{K}$*) genannt. Die Elemente von* V *werden als* Vektoren *und die Elemente von* $\mathbb{K}$ *als* Skalare *bezeichnet.*

Eine Anordnung von Skalaren $a_{ij} \in \mathbb{K}$ in m Zeilen und n Spalten wird $m \times n$*-Matrix* genannt. Auch hier gilt, dass die Menge aller $m \times n$-Matrizen mit der komponentenweise Matrizenaddition und der Multiplikation mit einem Skalar einen Vektorraum bilden.

Das Produkt $\mathbf{A} \cdot \mathbf{B}$ einer Matrix $\mathbf{A} \in \mathbb{K}^{m \times n}$ mit einer Matrix$\mathbf{B} \in \mathbb{K}^{n \times r}$ ist definiert als die Matrix $\mathbf{C} \in \mathbb{K}^{m \times r}$ mit den Koeffizienten

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} = a_{i1} b_{1j} + a_{i2} b_{2j} + \cdots + a_{in} b_{nj}$$

Die Multiplikation von Matrizen ist assoziativ und mit der Matrizenaddition distributiv. Sie ist jedoch nicht kommutativ, es ist also im Allgemeinen $\mathbf{A} \cdot \mathbf{B} \neq \mathbf{B} \cdot \mathbf{A}$. Die

Menge der quadratischen Matrizen bildet zusammen mit der Matrizenaddition und der Matrizenmultiplikation einen Ring mit Eins.

2.3.1 Vektornormen

Als *Norm* eines Vektors wird eine Abbildung eines Vektors auf einen Skalar bezeichnet, die die folgenden Bedingungen erfüllt.

Definition 2.5 (Vektornorm [TT08]). *Sei V ein Vektorraum über einem Körper $\mathbb{K}$ mit einer Funktion $\|\cdot\| : V \to \mathbb{K}$. Wenn*

$$\begin{aligned} \|\mathbf{u}\| \geq 0, \textit{ für } \mathbf{u} \neq \mathbf{0} && \textit{(Positivität)} \\ \|a\mathbf{u}\| = |a| \cdot \|\mathbf{u}\| && \textit{(Homogenität)} \\ \|\mathbf{u} + \mathbf{v}\| \leq \|\mathbf{u}\| + \|\mathbf{v}\| && \textit{(Dreiecksungleichung)} \end{aligned}$$

für alle $\mathbf{a}, \mathbf{b} \in V$ und $a \in \mathbb{K}$, wird $\|\cdot\|$ als Norm *bezeichnet.*

Für diese Arbeit relevant sind die Maximumsnorm und die Summennorm eines Vektors. Die Maximumsnorm oder Unendlichkeitsnorm gibt den Betrag der größten Komponente eines Vektors an.

Definition 2.6 (Maximumsnorm [KM03]). *Für einen Vektor $\mathbf{v} = (v_1, \ldots, v_n)$ nennt man*

$$\|\mathbf{v}\|_\infty := max\{|v_1|, \ldots, |v_n|\}$$

die Maximumsnorm $\|\cdot\|_\infty$ von $\mathbf{v}$.

Die Summennorm oder 1-Norm ist definiert als die Summe der Beträge der einzelnen Vektorkomponenten.

Definition 2.7 (Summennorm [KM03]). *Ist $\mathbf{v} = (v_1, \ldots, v_n)$ ein n-dimensionaler Vektor, dann ist die Summennorm $\|\cdot\|_1$ des Vektors definiert als*

$$\|\mathbf{v}\|_1 = \sum_{i=1}^{n} |v_i|.$$

2.3.2 Eigenwerte und Eigenvektoren

Ein Eigenvektor einer Matrix ist ein Vektor, der durch die Multiplikation mit der Matrix auf ein Vielfaches seiner selbst abgebildet wird. Der Faktor, um welchen der Vektor bei der Abbildung skaliert wird, wird als Eigenwert bezeichnet.

Definition 2.8 (Eigenwert und Eigenvektor [KM03]). *Sei* $\mathbf{A}$ *eine* $n \times n$*-Matrix mit Koeffizienten aus dem Körper* $\mathbb{K}$*. Ein Spaltenvektor* $\mathbf{v} \in \mathbb{K}^n$ *heisst* Eigenvektor *von* $\mathbf{A}$*, wenn* $\mathbf{v} \neq \mathbf{0}$ *und*

$$\mathbf{A}\mathbf{v} = f\mathbf{v}$$

mit einem Skalar $f \in \mathbb{K}$ *gilt, der dann wieder Eigenwert von* $\mathbf{A}$ *zum Eigenvektor* $\mathbf{v}$ *genannt wird.*

Haben zwei Matrizen $\mathbf{A}$ und $\mathbf{B}$ einen gemeinsamen Eigenvektor $\mathbf{v}$, so dass $\mathbf{A}\mathbf{v} = f\mathbf{v}$ und $\mathbf{B}\mathbf{v} = g\mathbf{v}$ gilt, gilt für die Summe und das Produkt der beiden Matrizen [GSW13]:

$$(\mathbf{A} + \mathbf{B}) \cdot \mathbf{v} = (f + g) \cdot \mathbf{v}$$

und

$$(\mathbf{A} \cdot \mathbf{B}) \cdot \mathbf{v} = (f \cdot g) \cdot \mathbf{v}$$

2.4 Stochastische Grundlagen

Um für ein kryptographisches Verfahren eine beweisbare Sicherheit gegen bekannte Angriffe garantieren zu können, muss die Verteilung des zufälligen Fehlers in den Verschlüsselungen bestimmte Kriterien erfüllen. Das im folgenden Abschnitt vorgestellte Learning with Errors Verfahren verwendet als Fehlerverteilung eine diskrete Verteilung über $\mathbb{Z}_q$ in Form einer Gaußschen Normalverteilung mit Erwartungswert 0.

Definition 2.9 (Diskrete Gauss-Verteilung [BV14]). *Für einen Skalierungsfaktor* $s > 0$ *ist die eindimensionale Gauß-Funktion* $\rho_s : \mathbb{R} \to (0, 1]$ *mit Erwartungswert* 0 *definiert als*

$$\rho_s(x) := e^{(-\pi|x|^2/s^2)}$$

Als stetige Gauss-Verteilung D_s *wird eine Wahrscheinlichkeitsverteilung mit einer zu* ρ_s *proportionalen Dichtefunktion bezeichnet. Die diskrete Gauss-Verteilung* $D_{\mathbb{Z}_q,s}$ *ist die diskrete Verteilung über* $\mathbb{Z}_q$*, deren Wahrscheinlichkeitsfunktion proportional zu* ρ_s *ist.*

Um diskret normalverteilte Stichproben $x \sim D_{\mathbb{Z}_q,s}$ zu erhalten, werden zunächst Samples von ρ_s erzeugt, die dann durch Multiplizieren mit q und Runden auf die nächste Ganzzahl modulo q über dem Bereich $\mathbb{Z}_q$ diskretisiert werden. Dabei müssen die Parameter so gewählt werden, dass die diskrete Gauss-Verteilung $D_{\mathbb{Z}_q,s}$ eine kontinuierliche Normalverteilung hinreichend gut approximiert.

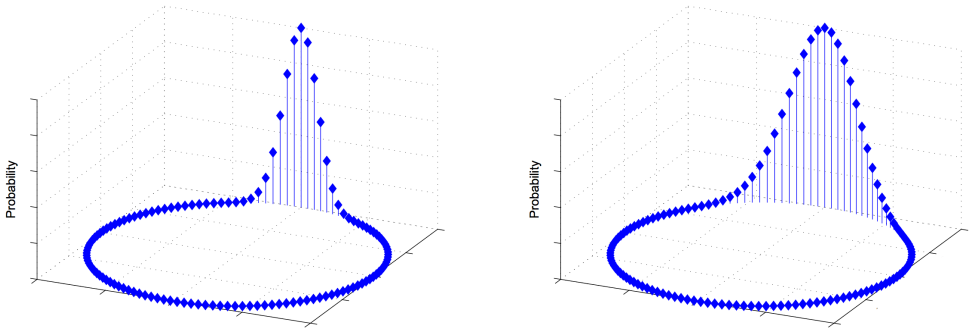

Abbildung 2.1: diskrete Gaußsche Wahrscheinlichkeitsverteilungen für $q = 127$ mit $\alpha = 0.05$ (rechts) und $\alpha = 0.1$ (rechts). Die Elemente von $\mathbb{Z}_q$ sind kreisförmig angeordnet [Reg09].

2.5 Gitterbasierte Kryptographie

Das in Kapitel 4 vorgestellte Verfahren basiert auf dem *Learning With Errors* Problem (LWE), einem Problem aus der gitterbasierten Kryptographie.

Ein Gitter bezeichnet eine diskrete, additive Untergruppe des $\mathbb{R}^m$. Anschaulicher lässt sich ein Gitter beschreiben als eine Menge von Punkten, die gitterförmig in einem m-dimensionalen Raum angeordnet sind, im $\mathbb{R}^3$ wären das beispielsweise die Atome eines Kristalls. Gitterbasierte kryptographische Verfahren nutzen Operationen in solchen Gittern, ihre Sicherheit basiert auf der Komplexität bestimmter Berechnungsprobleme in einem Gitter. Ein typisches Gitterproblem ist beispielsweise das Finden des nächsten Gitterpunktes von einer zufälligen Position im Raum.

Gitterbasierte Verfahren erfahren in der Kryptographie aktuell besondere Aufmerksamkeit, da vermutet wird, dass sie im Gegensatz zu den meisten aktuell verwendeten Verfahren auch auf einem Quantencomputer schwer bleiben. Es handelt sich also um ein Verfahren der Post-Quanten-Kryptographie.

2.5.1 Das Learning With Errors Problem

Die folgende Zusammenfassung orientiert sich stark an Regevs Beschreibung des Problems in [Reg10].

LWE wurde 2005 von O. Regev als Verallgemeinerung des Learning Parity With Noise Problems entwickelt [Reg05]. Vorteile des Systems sind seine Flexibilität und die Tatsache, dass sich die Sicherheit eines LWE Systems formal beweisen lässt. Regev und Peikert [Pei09] konnten zeigen, dass LWE mit bestimmten Parametern

mindestens so schwer ist, wie bestimmte gut untersuchte Gitter-Probleme. Damit bietet LWE eine Basis für beweisbar sichere und effiziente kryptografische Systeme.

Das LWE Problem bezeichnet das Problem, einen geheimen Vektor $\mathbf{s} \in \mathbb{Z}_q^n$ aus einer Menge von m linearen Gleichungen zu rekonstruieren, deren rechte Seite fehlerbehaftet ist. Ein Beispiel eines solchen Gleichungssystems wäre

$$\begin{aligned}
14s_1+15s_2+\ 5s_3+\ 2s_4 &\approx\ 8 \mod 17\\
13s_1+14s_2+14s_3+\ 6s_4 &\approx 16 \mod 17\\
16s_1+10s_2+13s_3+\ 1s_4 &\approx\ 3 \mod 17\\
10s_1+\ 4s_2+12s_3+16s_4 &\approx 12 \mod 17\\
\ 9s_1+\ 5s_2+\ 9s_3+\ 6s_4 &\approx\ 9 \mod 17\\
\ 3s_1+\ 6s_2+\ 4s_3+\ 5s_4 &\approx 16 \mod 17
\end{aligned}$$

wobei jede Gleichung korrekt ist bis auf einen kleinen additiven Fehler von ± 1 und das Ziel ist, den Vektor $\mathbf{s}$ herauszufinden.

Offensichtlich hängt die Schwierigkeit des Problems stark von der Verteilung des additiven Fehlers ab. Würde beim Erstellen der Gleichungen kein Fehlerterm hinzuaddiert, ließe sich das System effizient mit Hilfe des Gaußschen Eliminationsverfahrens lösen. Ist die Fehlerverteilung χ die diskrete Gleichverteilung auf $\mathbb{Z}_q$, wird es unmöglich, $\mathbf{s}$ zu rekonstruieren. Wenn χ eine diskrete Gaußsche Verteilung auf $\mathbb{Z}_q$ um 0 ist, so ist das diskrete LWE Problem schon bei relativ kleiner Standardabweichung von χ schwer.

Definition 2.10 (Learning With Errors [Reg09])**.** *Für positive Ganzzahlen n und $q \geq 2$, einen Vektor $\mathbf{s} \in \mathbb{Z}_q^n$ und eine Wahrscheinlichkeitsfunktion $\chi\colon \mathbb{Z}_q \to [0,1]$ ist $A_{\mathbf{s},\chi}$ die Verteilung auf $\mathbb{Z}_q^n \times \mathbb{Z}_q$, die gebildet wird, indem ein $\mathbf{a} \in \mathbb{Z}_q^n$ zufällig gleichverteilt und ein $e \in \mathbb{Z}_q$ davon unabhängig zufällig χ-verteilt gewählt wird und dann das Paar $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e)$ ausgegeben wird.*

Sind beliebig viele Samples von $A_{\mathbf{s},\chi}$ für ein unbekanntes, festes $\mathbf{s} \in \mathbb{Z}_q^n$ gegeben, löst ein Algorithmus das Learning with Errors Problem $LWE_{q,\chi}$ wenn er für jedes $\mathbf{s} \in \mathbb{Z}_q^n$ den Vektor $\mathbf{s}$ (mit hoher Wahrscheinlichkeit) ausgibt.

Die beschriebene Formulierung des Problems wird als LWE Suchproblem bezeichnet. Eine zweite Formulierung des Problems stellt das LWE Entscheidungsproblem (Decisional Learning with Errors oder DLWE) dar. Die DLWE Vermutung besagt, dass es keinen Polynomialzeit-Algorithmus gibt, der m Stichproben der beschriebenen

Verteilung $A_{\mathbf{s},\chi}$ von m Stichproben der Gleichverteilung über $\mathbb{Z}_q^n \times \mathbb{Z}_q$ unterschieden kann, für die ein $\mathbf{a} \in \mathbb{Z}_q^n$ und ein $b \in \mathbb{Z}_q$ zufällig gleichverteilt gewählt werden und das Paar $(\mathbf{a}, b)$ ausgegeben wird.

LWE basierte Verschlüsselung

Die Konstruktion eines auf dem LWE Problem basierenden Verschlüsselungssystems funktioniert wie folgt. Für die Verschlüsselung eines bits $m \in \{0, 1\}$ mit einem geheimen Schlüssel $\mathbf{s} \in \mathbb{Z}_q^n$ und einer Fehlerverteilung χ wird zufällig gleichverteilt ein Vektor $\mathbf{a} \in \mathbb{Z}_q^n$ sowie ein χ-verteilter Fehler e gewählt und daraus der Geheimtext

$$c = (\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle + 2e + m) \in \mathbb{Z}_q^n \times \mathbb{Z}_q \tag{2.1}$$

erzeugt, wobei alle Berechnungen in $\mathbb{Z}_q$ stattfinden [BV11a]. Die wichtigste Beobachtung für die Entschlüsselung der eigentlichen Nachricht ist, dass die beiden Komponenten, die die eigentliche Nachricht im Geheimtext überlagern – die geheime Maske $\langle \mathbf{a}, \mathbf{s} \rangle$ und die gerade Maske $2e$ – sich nicht gegenseitig beeinflussen. Der Geheimtext lässt sich entschlüsseln, indem die beiden Masken nacheinander entfernt werden. Im ersten Schritt berechnet der Entschlüsselungsalgorithmus $\langle \mathbf{a}, \mathbf{s} \rangle$, zieht das Ergebnis von $\mathbf{b}$ ab und erhält $[2e + m]_q$ als Ergebnis. Da e deutlich kleiner ist als q ist $[2e + m]_q = 2e + m$. Um jetzt noch die gerade Maske $2e$ zu entfernen, berechnet der Algorithmus $[2e + m]_2$ und erhält die ursprüngliche Nachricht m als Ergebnis.

Für eine korrekte Entschlüsselung muss der Fehlerterm $2e$ also gerade und im Vergleich zu q klein sein. Wählt man hier einen Fehler, der immer noch deutlich kleiner ist als q, aber teilbar durch eine größere Zahl als 2, lässt sich ein LWE Verschlüsselungssystem mit einem größeren Nachrichtenraum konstruieren. Wählt man beispielsweise einen durch 15 teilbaren Fehler $15e$ und passt den Entschlüsselungsalgorithmus so an, dass er nach Abzug von $\langle \mathbf{a}, \mathbf{s} \rangle$ das Ergebnis $[15e + m]_{15}$ ausgibt, lassen sich Nachrichten $m \in \{0, 1, \ldots, 14\}$ ver- und entschlüsseln.

Parameterwahl und Sicherheit

Eine LWE Instanz wird charakterisiert durch folgende Parameter. Der Modulus q legt den Restklassenring $\mathbb{Z}_q$ fest, in dem alle Berechnungen stattfinden. Der Dimensionsparameter n gibt die Länge des geheimen Schlüssels $\mathbf{s}$ an und bestimmt gleichzeitig die Größe der entstehenden Geheimtexte wie Gleichung 2.1 zeigt. Die Fehlerverteilung χ schließlich legt fest, wie der zufällige Fehler e verteilt ist. Fast alle aktuellen Untersuchungen zu LWE nutzen hier eine diskrete Gaußsche Verteilung

$D_{\mathbb{Z}_q,s}$.

Man geht davon aus, dass das LWE Problem mit sinnvoll gewählten Parametern sehr schwer ist, die besten bekannten Algorithmen zur Lösung des Problems haben exponentielle Laufzeiten in n [Reg05]. Hinweise auf die vermutete Schwere des Problems liefert zum einen die Nähe zum *Learning Parity with Noise* Problem, sowie die enge Verbindung von LWE zu Decodierungsproblemen aus der Codierungstheorie, welche ebenfalls als schwer gelten.

Weiterhin zeigt Regev in [Reg05] für eine Fehlerverteilung $\chi = D_{\mathbb{Z},\alpha\cdot q}$, mit $\alpha \cdot q \geq 2\sqrt{n}$ eine Worst Case Reduktion des LWE Suchproblems auf die beiden schweren Gitterprobleme GapSVP und approximate-SIVP, das Brechen eines LWE basierten Kryptosystems mit entsprechenden Parametern ist also so schwer wie das Lösen der Worst Case Instanz eines der beiden Probleme in einem n-dimensionalen Gitter. Regev führte die Reduktion mit Hilfe eines Quantenalgorithmus durch, er konnte also nur beweisen, dass aus einem Polynomialzeitalgorithmus, der das LWE Problem löst, die Existenz eines Quantenalgorithmus für GapSVP und approximate-SIVP folgt. Peikert konnte diesen Zusammenhang 2009 für ähnliche Parameter und ausreichend große Moduli q auch für klassische Algorithmen zeigen [Pei09].

2.6 Boolesche Schaltkreise

In homomorphen Verschlüsselungssystemen werden die homomorph zu berechnenden Funktionen in der Regel als *Boolesche Schaltkreise* dargestellt. Ein boolescher Schaltkreis ist definiert über einer Basis wie $(\mathsf{NOT}, \mathsf{AND}, \mathsf{OR})$, also einer Menge logischer Operationen, aus denen sich alle anderen logischen Funktionen konstruieren lassen. Er ist definiert als endlicher, gerichteter azyklischer Graph, dessen Knoten einer elementaren logischen Operation, einem Eingabe- oder einem Ausgabebit entsprechen. Durch die Verdrahtung zu einem Schaltkreis kann jede boolesche Funktion $F : \{0,1\}^n \to \{0,1\}^m$ mit $m, n > 0$ berechnet werden.

Die *Größe* eines Schaltkreises C ist die Anzahl seiner Gatter, die *Tiefe* von C ist definiert als die Länge des längsten gerichteten Pfades im Graph.

2.6.1 Branching Programme

Eine alternatives Modell zur Berechnung Boolescher Funktionen stellen *Branching Programme* dar. Die Auswertung einer Funktion als Branching Programm statt als Schaltkreis wirkt sich im in Kapitel 4 vorgestellten System günstig auf das Fehlerwachstum während der homomorphen Berechnung aus.

Definition 2.11 (Branching Program [Weg87]). *Ein Branching Program oder binäres Entscheidungsdiagramm ist ein gerichteter, azyklischer Graph, aufgebaut aus einer Wurzel, inneren Knoten und zwei mit den booleschen Konstanten* 0 *und* 1 *beschrifteten Blättern. Jeder innere Knoten wird mit einer booleschen Variable x_i bezeichnet und besitzt genau zwei ausgehende mit* 0 *und* 1 *beschriftete Kanten.*

Die Berechnung einer Funktion $f(x_1, \ldots, x_n)$ mit Hilfe eines Branching Programs für eine binäre Eingabe beginnt an der Wurzel und folgt dann für jeden Knoten x_i der ausgehenden Kante mit dem Wert des i-ten Eingangsbits bis zu einem der beiden Blätter des Graphen, welcher das Ergebnis von $f(x_1, \ldots, x_n)$ angibt.

Eine Sonderform stellen *Permutation Branching Programs* dar. In einem Permutation Branching Program bilden die ausgehenden 1-Kanten pro Ebene eine zyklische Permutation, die 0-Kanten bilden die identische Permutation. Abbildung 2.2 zeigt ein Beispiel eines solchen Programms.

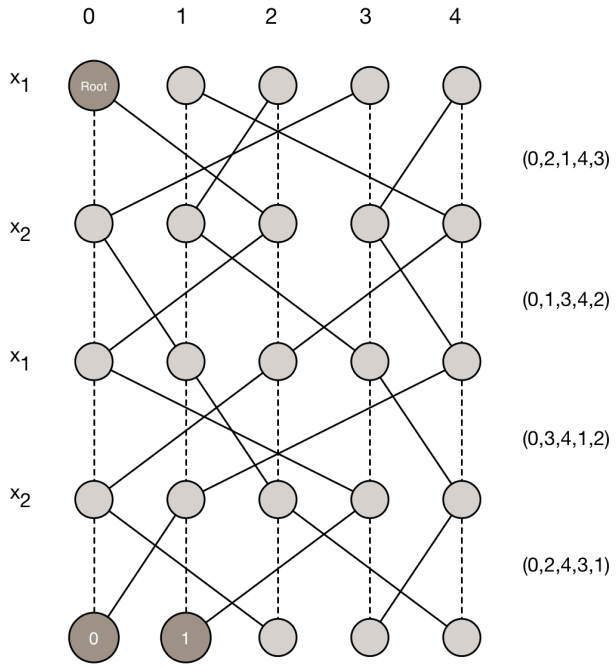

Abbildung 2.2: Permutation Branching Program der Breite 5 [Bar86] für eine Funktion $f\colon \{0,1\}^2 \to \{0,1\}$. Kanten für $x_i = 1$ sind durchgezogen, für $x_i = 0$ gestrichelt dargestellt. Um $f(x_1, x_2)$ auszuwerten, beginnt man am ROOT-Knoten und folgt den dem Variablenwert entsprechenden Kanten. Das Beispiel berechnet $f = \mathsf{AND}(x_1, x_2)$, die Eingabe $x_1 = x_2 = 1$ wertet das Programm zu 1 aus, alle übrigen Eingaben zu 0.

Definition 2.12 ([Weg87]). *Die* Größe *eines Branching Programs bezeichnet die Anzahl der inneren Knoten des Graphs, die* Tiefe *gibt die Länge des längsten Pfades an. Werden die Knoten in einer Folge von Ebenen angeordnet, so dass Kanten nur von einer Ebene in die*

nächste führen, bezeichnet die Breite *des Branching Programs die maximale Anzahl von Knoten in einer Ebene.*

Kapitel 3

Homomorphe Verschlüsselung

Homomorphe Verschlüsselung überträgt den in Kapitel 2.2.1 vorgestellten Homomorphiebegriff auf ein kryptographisches System.

Das folgende Kapitel definiert zunächst homomorphe Verschlüsselung und die Anforderungen an ein solches System. Anschliessend wird exemplarisch am Beispiel von Gentrys erstem voll homomorphen Verschlüsselungssystem [Gen09] das grundsätzliche Vorgehen zur Konstruktion eines FHE Systems beschrieben.

3.1 Verschlüsselungsverfahren

Ein Verschlüsselungssystem $\mathcal{E} = (\mathsf{Gen}_\mathcal{E}, \mathsf{Enc}_\mathcal{E}, \mathsf{Dec}_\mathcal{E})$ ist ein 3-Tupel bestehend aus Algorithmen zur Schlüsselerzeugung, Verschlüsselung und Entschlüsselung von Nachrichten. Diese Algorithmen müssen effizient, also in Polynomialzeit arbeiten, wobei die Laufzeit polynomiell von einem Sicherheitsparameter λ abhängt. Dieser legt das gewünschte Sicherheitsniveau des Systems fest, oft entspricht λ der Bitlänge der verwendeten Schlüssel. Die Ver- und Entschlüsselungsfunktionen $\mathsf{Enc}_\mathcal{E}$, und $\mathsf{Dec}_\mathcal{E}$ sind nur für Nachrichten aus einem festgelegten Nachrichtenraum $\mathcal{M}$ definiert. Die in dieser Arbeit vorgestellten homomorphen Verfahren verschlüsseln bitweise, hier gilt also, wenn nicht anders dargestellt, $\mathcal{M} = \{0, 1\}$. Codierungen längerer Nachrichten bestehen aus Aneinanderreihungen einzeln verschlüsselter Bits.

In einem *symmetrischen* Verschlüsselungssystem mit Sicherheitsparameter λ erzeugt $\mathsf{Gen}_\mathcal{E}$ einen von λ abhängigen geheimen Schlüssel sk, der sowohl von $\mathsf{Enc}_\mathcal{E}$ als auch von $\mathsf{Dec}_\mathcal{E}$ genutzt wird, um eine Nachricht auf einen Geheimtext abzubilden und umgekehrt.

In einem *asymmetrischen* Verschlüsselungssystem benötigt die Funktion zur Schlüsselerzeugung $\mathsf{Enc}_\mathcal{E}$ ebenfalls den Sicherheitsparameter λ als Eingabe und erzeugt

ein Paar aus entsprechend großem öffentlichen und geheimen Schlüssel (pk, sk).

$$\mathsf{Gen}_{\mathcal{E}}(\lambda) = (pk, sk)$$

Die Verschlüsselungsfunktion $\mathsf{Enc}_{\mathcal{E}}$ berechnet aus dem öffentlichen Schlüssel pk und einer Nachricht $\mu \in \mathcal{M}$ einen Geheimtext $\mathbf{c} \in \mathcal{C}$.

$$\mathsf{Enc}_{\mathcal{E}}(pk, \mu) = \mathbf{c}$$

Die Entschlüsselungsfunktion $\mathsf{Dec}_{\mathcal{E}}$ schließlich berechnet mit Hilfe des geheimen Schlüssels sk aus der verschlüsselten Nachricht $\mathbf{c}$ wieder einen Klartext μ.

$$\mathsf{Dec}_{\mathcal{E}}(sk, \mathbf{c}) = \mu$$

Dabei ergeben sich zwei Anforderungen an ein Verschlüsselungsverfahren, es muss *korrekt* und *sicher* sein.

3.1.1 Korrektheit

Offensichtlich ist ein Verschlüsselungsverfahren dann korrekt, wenn jeder aus einer Nachricht μ erzeugte Geheimtext $\mathbf{c}$ wieder zur ursprünglichen Nachricht μ entschlüsselt werden kann. Es muss also gelten:

Definition 3.1 (Korrektheit eines Public Key Verfahrens [Gen10])**.** *Ein asymmetrisches Verschlüsselungsverfahren ist* korrekt*, wenn gilt:*

$$\forall \mu \in \{0,1\}^n, \forall (pk, sk) \leftarrow \mathsf{Gen}_{\mathcal{E}}(\lambda) \colon \mathsf{Dec}_{\mathcal{E}}(sk, \mathsf{Enc}_{\mathcal{E}}(pk, \mu)) = \mu$$

3.1.2 Sicherheit

In einem sicheren kryptographischen System soll kein Angreifer das Verfahren in Polynomialzeit mit einer signifikanten Wahrscheinlichkeit brechen können, wenn die verwendeten Parameter und Schlüssel zufällig gewählt wurden. Damit handelt es sich um ein Average Case Konzept: das mathematische Problem, welches die kryptologische Sicherheit des Systems garantiert, muss im Average Case schwer sein. Ein Problem, welches lediglich im Worst Case schwer ist, reicht für kryptographische Zwecke nicht aus. In einem solchen Fall kann lediglich garantiert werden, dass es keinen Angreifer gibt, der das Verfahren für *alle* denkbaren Parameter und Schlüssel in Polynomialzeit brechen kann.

Zur genauen Analyse der Sicherheitseigenschaften eines Systems definiert die Kryptoanalyse verschiedene Sicherheitsbegriffe, die angeben, welches Sicherheitsniveau

ein System bei einer bestimmten Art von Angriff auf eine bestimmtes Sicherheitsziel aufweist. Die Mindestanforderung an ein sicheres asymmetrisches Verschlüsselungsverfahren ist die Forderung nach semantischer Sicherheit.

Definition 3.2 (Semantische Sicherheit [Gen10])**.** *Ein Verschlüsselungsverfahren ist* semantisch sicher*, wenn es keinen Polynomialzeit-Algorithmus gibt, mit dessen Hilfe ein Angreifer aus einem Geheimtext und dem zugehörigen öffentlichen Schlüssel Informationen über die verschlüsselte Nachricht ableiten kann.*

Eine Verstärkung dieser Anforderung stellt die Forderung nach *Ununterscheidbarkeit von Geheimtexten* oder *IND-CPA (ciphertext-indistinguishability under chosen plaintext attacks)* dar. Ein Angreifer darf einem Geheimtext **c**, der entweder die Nachricht μ_0 oder μ_1 codiert, den zugrundeliegenden Klartext auch dann nicht zuordnen können, wenn er μ_0 und μ_1 selbst gewählt hat.

Ein *deterministisches* Verschlüsselungsverfahren, das eine Nachricht immer auf denselben Geheimtext abbildet, kann also nicht semantisch sicher sein. Ein Angreifer kann leicht feststellen, welchen Klartext **c** codiert, indem er die selbstgewählten Klartexte mit Hilfe des öffentlichen Schlüssels verschlüsselt und das Ergebnis mit **c** vergleicht. Ein semantisch sicheres Kryptosystem muss *probabilistisch* sein, es muss unterschiedliche Verschlüsselungen einer gegebene Nachricht μ geben und $\mathsf{Enc}_{\mathcal{E}}(\mu)$ muss zufällig eine davon auswählen. Offensichtlich nutzt ein probabilistisches System nur für Schlüsselerzeugung und Verschlüsselung probabilistische Algorithmen während die Entschlüsselung deterministisch ablaufen muss.

Circular Security

Ein weiterer Sicherheitsbegriff, der in Bezug auf homomorphe Verschlüsselung eine Rolle spielt, ist die Forderung nach *Circular Security*. Für die Konstruktion eines voll homomorphen Verschlüsselungssystems gibt es aktuell unterschiedliche Ansätze, einer davon ist die Anwendung des in Kapitel 3.3.2 beschriebenen *Bootstrapping Theorems* [Gen09]. Dazu muss zusätzlich zum öffentlichen Schlüssel pk eines Systems eine mit diesem Schlüssel verschlüsselte Version des geheimen Schlüssels $\mathsf{Enc}_{\mathcal{E}}(pk, sk)$ veröffentlicht werden.

Ein System, dessen Sicherheit auch in diesem Fall noch gewährleistet ist, in dem der mit dem eigenen öffentlichen Schlüssel verschlüsselte Secret Key eines Verschlüsselungssystems also keine zusätzlichen Informationen preisgibt, bezeichnet man als *circular secure* [Gen09]. In verschiedenen Arbeiten zum Thema FHE wird vermutet, dass sich aus der semantischen Sicherheit eines bitweise verschlüsselnden Systems dessen Circular Security folgern lässt, was Ron Rothblum 2012 widerlegen konnte. Er konstruierte ein asymmetrisches Verschlüsselungssystem, welches semantisch

sicher ist, aber nicht über Circular Security verfügt [Rot13]. Er schwächte die Vermutung dahingehend ab, dass jedes bitweise operierende System, welches sicher gegen adaptive chosen-ciphertext Angriffe (CCA) ist, auch über Circular Security verfügt.

3.2 Homomorphe Verschlüsselung

Verschlüsselung beschreibt allgemein eine Abbildung von Klartext-Nachrichten aus einem definierten Nachrichtenraum $\mathcal{M}$ auf Geheimtexte aus einem Geheimtextraum $\mathcal{C}$. In einem *homomorphen Verschlüsselungssystem* sind zusätzlich Verknüpfungen auf den Nachrichten beziehungsweise den Geheimtexten definiert, so dass die Menge der Nachrichten mit den zugehörigen Verknüpfungen $(\mathcal{M}, +, \cdot)$ und die Menge der Geheimtexte entsprechend als $(\mathcal{C}, \oplus, \otimes)$ algebraische Strukturen darstellen. Die Ver- und Entschlüsselungsfunktion beschreiben homomorphe Abbildungen zwischen diesen beiden Strukturen, für ein homomorphes Verschlüsselungssystem $\mathcal{E}$ und Nachrichten $\mu_1, \mu_2 \in \mathcal{M}$ gilt also:

$$\mathsf{Enc}_{\mathcal{E}}(\mu_1) \oplus \mathsf{Enc}_{\mathcal{E}}(\mu_2) = \mathsf{Enc}_{\mathcal{E}}(\mu_1 + \mu_2)$$
$$\mathsf{Enc}_{\mathcal{E}}(\mu_1) \otimes \mathsf{Enc}_{\mathcal{E}}(\mu_2) = \mathsf{Enc}_{\mathcal{E}}(\mu_1 \cdot \mu_2)$$

Zur homomorphen Auswertung von Funktionen auf den verschlüsselten Daten existiert in einem homomorphen System ein vierter Algorithmus $\mathsf{Eval}_{\mathcal{E}}(evk, f, \mu_1, \mu_2, \dots)$. Dieser benötigt als Eingabe einen speziellen öffentlichen Schlüssel *evk*, die Beschreibung der auszuwertenden Funktion f sowie die Geheimtexte, die die Eingabewerte verschlüsseln. Als Ausgabe wird ein Geheimtext berechnet, der sich zu $f(\mu_1, \mu_2, \dots)$ entschlüsseln lässt.

Definition 3.3 (Homomorphe Verschlüsselung [Vai11])**.** *Ein homomorphes Verschlüsselungssystem $\mathcal{E} = (\mathsf{Gen}_{\mathcal{E}}, \mathsf{Enc}_{\mathcal{E}}, \mathsf{Dec}_{\mathcal{E}}, \mathsf{Eval}_{\mathcal{E}})$ ist ein 4-Tupel der folgenden Polynomialzeit-Algorithmen.*

Schlüsselerzeugung *Der Algorithmus $(pk, evk, sk) \leftarrow \mathsf{Gen}_{\mathcal{E}}(\lambda)$ erwartet den Sicherheitsparameter λ als Input und erzeugt einen öffentlichen Schlüssel pk für die Verschlüsselung, einen öffentlichen Schlüssel evk, welcher für die Auswertung von Funktionen unter Verschlüsselung benötigt wird, sowie einen privaten Schlüssel sk zur Entschlüsselung.*

Verschlüsselung *Der Algorithmus $\mathbf{c} \leftarrow \mathsf{Enc}_{\mathcal{E}}(pk, \mu)$ akzeptiert den öffentlichen Schlüssel pk und eine einzelne Bit-Nachricht $\mu \in \{0, 1\}$ und erzeugt als Ausgabe einen Geheimtext* $\mathbf{c}$*.*

Entschlüsselung *Der Algorithmus* $\mu \leftarrow \mathsf{Dec}_{\mathcal{E}}(sk, \mathbf{c})$ *akzeptiert den geheimen Schlüssel sk und einen Geheimtext* $\mathbf{c}$ *und gibt eine Nachricht* $\mu \in \{0,1\}$ *aus.*

homomorphe Berechnung *Der Algorithmus* $(\mathbf{c}_{f1}, \ldots, \mathbf{c}_{fm}) \leftarrow \mathsf{Eval}_{\mathcal{E}}(evk, f, \mathbf{c}_1, \ldots, \mathbf{c}_l)$ *erwartet den Schlüssel zur Funktionsauswertung evk, die Beschreibung einer Booleschen Funktion* $f\colon \{0,1\}^l \rightarrow \{0,1\}^m$*, sowie eine Menge von l Geheimtexten* $\mathbf{c}_1, \ldots, \mathbf{c}_l$ *mit* $\mathbf{c}_i \leftarrow \mathsf{Enc}_{\mathcal{E}}(pk, \mu_i)$ *und gibt eine Menge von Geheimtexten* $(\mathbf{c}_{f1}, \ldots, \mathbf{c}_{fm})$ *aus.*

Die Ausgabe von $\mathsf{Eval}_{\mathcal{E}}$ lässt sich wieder zum Ergebnis der Funktion f, angewendet auf die umverschlüsselten Eingangsparameter $\mu_1, \ldots, \mu_l$, entschlüsseln, ohne weitere Informationen über $\mu_1, \ldots, \mu_l$ oder über f preiszugeben [Gen10].

$$\mathsf{Dec}_{\mathcal{E}}(sk, \mathsf{Eval}_{\mathcal{E}}(evk, f, \mathsf{Enc}_{\mathcal{E}}(pk, \mu_1), \ldots, \mathsf{Enc}_{\mathcal{E}}(pk, \mu_l))) = f(\mu_1, \ldots, \mu_l)$$

Die zu berechnende Funktion $f\colon \{0,1\}^l \rightarrow \{0,1\}^m$ wird in der Regel als Boolescher Schaltkreis aufgebaut aus AND, OR und NOT-Gattern mit l Eingangs- und m Ausgangsgattern dargestellt. Vorteil dieser Darstellungsart ist, dass sie einerseits eine schrittweise Auswertung der gewünschten Funktion ermöglicht und andererseits vergleichbare Aussagen über deren Komplexität zulässt.

3.2.1 Korrektheit

Um die Korrektheit eines homomorphen Verschlüsselungssystems zu garantieren, müssen die bereits formulierten Standardanforderungen an ein korrektes asymmetrisches Verschlüsselungssystem erweitert werden. Zusätzlich zur korrekten Ver- und Entschlüsselung muss auch die Korrektheit der homomorphen Evaluation sichergestellt sein. Ein System ist homomorph in Bezug auf eine Familie von Schaltkreisen, wenn folgendes gilt:

Definition 3.4 (C-Homomorphismus [Vai11])**.** *Ein Verschlüsselungssystem* $\mathcal{E}$ *ist C-homomorph für eine Familie boolescher Schaltkreise* $\mathcal{C} = \{\mathcal{C}_\eta\}_{\eta \in \mathbb{N}}$*, wenn für jedes* $C \in \mathcal{C}$ *mit l Eingängen und entsprechenden Eingangsparametern* $\mu_1, \ldots, \mu_l \in \{0,1\}$ *gilt, dass*

$$\mathsf{Dec}_{\mathcal{E}}(sk, \mathsf{Eval}_{\mathcal{E}}(evk, C, \mathbf{c}_1, \ldots, \mathbf{c}_l)) = C(\mu_1, \ldots, \mu_l)$$

für $(pk, sk) \leftarrow \mathsf{Gen}_{\mathcal{E}}(\lambda)$ *und* $\mathbf{c}_i \leftarrow \mathsf{Enc}_{\mathcal{E}}(sk, \mu_i)$*.*

Mit der bisherigen Definition eines HE-Schemas lassen sich noch relativ nutzlose homomorphe Systeme konstruieren. Beispielsweise könnte man ein beliebiges Verschlüsselungssystem in ein homomorphes System überführen, indem man

$\mathsf{Eval}_{\mathcal{E}}(evk, f, \mathbf{c}_1, \ldots, \mathbf{c}_t)$ als Identitätsfunktion definiert, so dass es $\mathbf{c} = (f, \mathbf{c}_1, \ldots, \mathbf{c}_t)$ als Aneinanderreihung der Eingabeparameter ausgibt, ohne weitere Berechnungen vorzunehmen. $\mathsf{Dec}_{\mathcal{E}}(sk, \mathbf{c})$ wird insofern angepasst, dass es zur Decodierung von $\mathbf{c}$ zunächst $\mathbf{c}_1, \ldots, \mathbf{c}_t$ entschlüsselt und danach f auf die Klartexte $\mu_1, \ldots, \mu_t$ anwendet. Offensichtlich lässt sich mit dieser trivialen Lösung das eigentliche Ziel homomorpher Verschlüsselung, das *Auslagern* von Berechnungen unter Verschlüsselung nicht erreichen. Es ergibt sich also als zusätzliche Anforderung an das System, dass die Entschlüsselung einer Ausgabe von $\mathsf{Eval}_{\mathcal{E}}$ nicht komplexer sein darf, als die Entschlüsselung eines frisch verschlüsselten Geheimtextes.

Haben die von $\mathsf{Enc}_{\mathcal{E}}$ und $\mathsf{Eval}_{\mathcal{E}}$ erzeugten Geheimtexte zusätzlich das gleiche Format, spricht man von stark homomorpher Verschlüsselung.

Definition 3.5 (Stark homomorphe Verschlüsselung [Vai11])**.** *Sei $\mathcal{E}$ ein für Schaltkreise der Klasse $\mathcal{C}$ C-homomorphes Verschlüsselungssystem mit Sicherheitsparameter λ. Man bezeichnet $\mathcal{E}$ als* stark homomorph*, wenn $\forall\lambda, \forall C \in \mathcal{C}, \forall\mu_i \in \{0,1\}, 1 \leq i \leq t$ und $\forall \mathbf{c}_i \leftarrow \mathsf{Enc}_{\mathcal{E}}(pk, \mu_i)$ ein homomorph ausgewerteter Geheimtext $\mathbf{c}_{eval} \leftarrow \mathsf{Eval}_{\mathcal{E}}(C, \mathbf{c}_1, \ldots, \mathbf{c}_t)$ nicht von einem frisch verschlüsselten Geheimtext $c_i \leftarrow \mathsf{Enc}_{\mathcal{E}}(pk, \mu_i)$ zu unterscheiden ist.*

Einige der vorgestellten teilweise homomorphe Verschlüsselungssysteme sind in der Tat stark homomorph in Bezug auf ihre homomorph durchführbare algebraische Addition, so zum Beispiel das Goldwasser-Micalli oder El Gamal System. Für viele Anwendungen genügt aber die schwächere Forderung nach Kompaktheit der homomorph berechneten Geheimtexte. Diese stellt keine Anforderungen an das Format der Geheimtexte nach homomorpher Evaluation, sondern verlangt lediglich, dass deren Länge weder von der Anzahl der Eingänge, noch von der Komplexität des ausgewerteten Schaltkreises abhängt.

Definition 3.6 (Schwach homomorphe Verschlüsselung / Kompaktheit [Vai11])**.** *Ein homomorphes Verschlüsselungsschema $\mathcal{E}$ mit Sicherheitsparameter λ ist kompakt, wenn ein polynomiell in λ beschränktes $s = s(\lambda)$ existiert, so dass die Länge der Ausgabe von $\mathsf{Eval}_{\mathcal{E}}$ höchstens s Bits beträgt, unabhängig von der ausgewerteten Funktion f und der Anzahl ihrer Eingabewerte.*

Während die bisher formulierten Definitionen homomorpher Verschlüsselung Einschränkungen für die auszuwertenden Schaltkreisfamilien $\mathcal{C}$ enthalten, spricht man von voll homomorpher Verschlüsselung, wenn ein System beliebige Funktionen auf den verschlüsselten Daten auswerten kann.

Definition 3.7 (Voll homomorphe Verschlüsselung (FHE) [Vai11])**.** *Ein Verschlüsselungsschema $\mathcal{E}$ ist* voll homomorph*, wenn es kompakt und homomorph ist für die Klasse aller Booleschen Schaltkreise.*

Dass voll homomorphe Verschlüsselung theoretisch möglich ist, zeigte Craig Gentry 2009 in seiner Dissertation [Gen09] (siehe Kapitel 3.3), allerdings sind existierende FHE Systeme so komplex - sowohl in Bezug auf die Laufzeit der Algorithmen als auch auf den benötigten Speicherplatz - dass ein praktischer Einsatz voll homomorpher Verschlüsselung in der aktuellen Form nicht in Frage kommt. Für viele Anwendungen ist es nicht nötig, *beliebige* Berechnungen auf den verschlüsselten Daten auszuführen da Art und Tiefe der benötigten Schaltkreise feststehen. Damit bleiben Schemas, welche nur bestimmte Klassen von Schaltkreisen homomorph auswerten können, dafür aber eine geringere Komplexität aufweisen, weiterhin interessant. Solche Systeme werden *Leveled Fully Homomorphic Encryption Schemes* genannt, die Systemparameter und die Schlüsselgrößen eines solchen Systems hängen nicht mehr nur vom Sicherheitsparameter λ ab, sondern zusätzlich von der gewünschten Tiefe L der Schaltkreise, die das System homomorph auswerten soll.

Definition 3.8 (Leveled Fully Homomorphic Encryption (LFHE) [Vai11])**.** *Ein* leveled fully homomorphic encryption scheme *ist ein homomorphes Verschlüsselungssystem, in dem die Funktion zur Schlüsselerzeugung* $(pk, evk, sk) \leftarrow \mathsf{Gen}_{\mathcal{E}}(\lambda, L)$ *einen zusätzlichen Eingangsparameter L mit* $L \in \mathbb{N}$ *erhält und das resultierende Schema* $\mathcal{E}$ *C-homomorph ist für alle Booleschen Schaltkreise der Tiefe L. Die Größe der erzeugten Geheimtexte ist nur von* λ *abhängig und bleibt unabhängig von L.*

3.3 Gentrys Blueprint

Craig Gentry entwickelte in seiner Dissertation [Gen09] nicht nur das erste voll homomorphe Verschlüsselungssystem sondern beschrieb auch ein allgemeines Vorgehen, um ein solches System zu konstruieren. Ausgehend von Gentrys so genanntem *Blueprint* entstanden eine Reihe weiterer FHE Systeme, aufbauend auf besser untersuchten Sicherheitsannahmen, mit einfacheren oder effizienteren Algorithmen.

Gentrys Konstruktion besteht aus drei Komponenten: einem *somewhat* homomorphen Verschlüsselungssystem, welches eine begrenzte Familie von Funktionen auswerten kann, einer von Gentry als *Bootstrapping* bezeichneten Methode, mit deren Hilfe ein ausreichend mächtiges homomorphes Schema in ein voll homomorphes Schema überführt wird und einer Methode um das somewhat homomorphe System zu einem Schema zu erweitern, in dem Bootstrapping möglich ist.

3.3.1 Somewhat Homomorphic Encryption

Der erste Schritt in Gentrys Blueprint besteht in der Entwicklung eines Verschlüsselungssystems, welches Polynome bis zu einem bestimmten (kleinen) Grad homo-

morph auswerten kann. Gentry spricht von einem *Somewhat Homomorphic Encryption Scheme (SWHE)*. Die begrenzte Anzahl der möglichen aufeinanderfolgenden homomorphen Berechnungen folgt aus der probabilistischen Verschlüsselung, die durch kleine zufällige Fehler in den Geheimtexten erreicht wird. Werden Geheimtexte miteinander multipliziert oder addiert, erhöht sich auch der Betrag des Fehlers in den Verschlüsselungen, so dass nur eine begrenzte Anzahl an homomorphen Operationen möglich ist, bevor das Rauschen so groß wird, dass die korrekte Entschlüsselung der eigentlichen Nachricht unmöglich wird.

3.3.2 Bootstrapping

Bootstrapping basiert auf der Idee, dass es in einem homomorphen System möglich ist, für einen Geheimtext $\mathbf{c}$ die eigene Entschlüsselungsfunktion mit einer verschlüsselten Version des geheimen Schlüssels sk homomorph auszuwerten. Das Ergebnis ist eine aktualisierte Version des Geheimtextes $\mathbf{c}$ mit reduziertem Rauschen. Wenn das System zusätzlich in der Lage ist, mindestens eine weitere Operation auszuwerten, lässt sich daraus ein voll homomorphes System entwickleln [Gen10].

Definition 3.9 (Bootstrappable Encryption Scheme [Vai11])**.** *Das Verschlüsselungssystem $\mathcal{E}$ sei C-homomorph für Schaltkreise der Klasse $\mathcal{C}$ und f_{add} und f_{mult} seien die erweiterten Entschlüsselungsfunktionen von $\mathcal{E}$, definiert als*

$$f_{add}^{c_1,c_2}(sk) = \mathsf{XOR}(\mathsf{Dec}_{\mathcal{E}}(sk, \mathbf{c}_1), \mathsf{Dec}_{\mathcal{E}}(sk, \mathbf{c}_2))$$
$$f_{mult}^{c_1,c_2}(sk) = \mathsf{AND}(\mathsf{Dec}_{\mathcal{E}}(sk, \mathbf{c}_1), \mathsf{Dec}_{\mathcal{E}}(sk, \mathbf{c}_2))$$

$\mathcal{E}$ ist bootstrappable*, wenn*

$$\{f_{add}^{c_1,c_2}, f_{mult}^{c_1,c_2}\}_{c_1,c_2} \subseteq \mathcal{C}.$$

Die Funktionsweise von Bootstrapping lässt sich folgendermassen erklären [Gen10]: In einem Verschlüsselungssystem $\mathcal{E}$ sei $C_{\mathsf{Dec}_{\mathcal{E}}}$ die Entschlüsselungsfunktion des Systems, ausgedrückt als Schaltkreis. Das System sei bootstrapable, also in der Lage die eigene Entschlüsselungsfunktion homomorph auszuwerten. Der Geheimtext $\mathbf{c}_1 = \mathsf{Enc}_{\mathcal{E}}(pk_1, \mu)$ sei eine Verschlüsselung des Bits μ unter dem öffentlichen Schlüssel pk_1 und $\overline{sk_1}$ sei ein Vektor von Geheimtexten, welche die Bits von sk_1 unter einem neuen öffentlichen Schlüssel des Systems pk_2 als $\mathsf{Enc}_{\mathcal{E}}(pk_2, sk_{1j})$ verschlüsseln. Folgender Algorithmus erzeugt dann eine neue Verschlüsselung von μ unter pk_2.

Funktion Recrypt($pk_2, C_{\mathsf{Dec}_\mathcal{E}}, \overline{sk_1}, \mathbf{c}_1$)

Eingabe : $pk_2 \leftarrow \mathsf{Gen}_\mathcal{E}(\lambda), C_{\mathsf{Dec}_\mathcal{E}}, \overline{sk_1}, \mathbf{c}_1 \leftarrow \mathsf{Enc}_\mathcal{E}(pk_1, \mu)$

1 Erzeuge $\overline{\mathbf{c}_1}$ als $\mathsf{Enc}_\mathcal{E}(pk_2, c_{1_j})$ für jedes Bit c_{1_j} von $\mathbf{c}_1$;

Ausgabe : $\mathbf{c}_1' \leftarrow \mathsf{Eval}_\mathcal{E}(pk_2, C_{\mathsf{Dec}_\mathcal{E}}, \overline{sk_1}, \overline{\mathbf{c}_1})$

Der Schaltkreis $C_{\mathsf{Dec}_\mathcal{E}}$ hat Eingänge für die Bits eines geheimen Schlüssels und die eines Geheimtextes. In $\mathsf{Recrypt}_\mathcal{E}$ wird $\mathsf{Eval}_\mathcal{E}$ mit den Bits von sk_1 und $\mathbf{c}_1$ aufgerufen, allerdings homomorph unter Verschlüsselung mit pk_2. Das Ergebnis ist entsprechend die mit pk_2 verschlüsselte Ausgabe von $\mathsf{Dec}_\mathcal{E}(sk_1, \mathbf{c}_1) = \mu$. Für das Rauschen in den beiden Geheimtexten $\mathbf{c}_1$ und $\mathbf{c}_1'$ gilt, dass die Auswertung von $C_{\mathsf{Dec}_\mathcal{E}}$ das Rauschen der ursprünglichen Verschlüsselung c_1 unter pk_1 entfernt, während die Auswertung des Schaltkreises unter pk_2 neues Rauschen erzeugt. Solange dieser neu hinzukommende Fehler kleiner ist, als der durch $\mathsf{Dec}_\mathcal{E}(sk_1, \mathbf{c}_1)$ entfernte, hat $\mathsf{Recrypt}_\mathcal{E}$ das Rauschen im Geheimtext reduziert.

Um nicht nur eine neue Verschlüsselung derselben Nachricht zu erhalten, sondern eine tatsächliche Operation auf den unterliegenden Nachrichten auszuführen, muss $\mathcal{E}$ den Schaltkreis $C_{\mathsf{Dec}_\mathcal{E}}$ erweitert um ein zusätzliches Gatter homomorph auswerten können. Um beispielsweise eine Addition durchzuführen, sei dieser erweiterte Schaltkreis $C_{\mathsf{Dec}_\mathcal{E}, Add}$. Wenn $\mathbf{c}_1$ und $\mathbf{c}_2$ Verschlüsselungen der Nachrichten μ_1 und μ_2 unter pk_1 sind, und $\overline{\mathbf{c}_1}$ und $\overline{\mathbf{c}_2}$ erzeugt werden wie beschrieben, dann gilt

$$c_{Add} \leftarrow \mathsf{Eval}_\mathcal{E}(pk_2, C_{\mathsf{Dec}_\mathcal{E}, Add}, \overline{sk_1}, \overline{\mathbf{c}_1}, \overline{\mathbf{c}_2}) = \mathsf{Enc}_\mathcal{E}(pk_2, \mu_1 + \mu_2)$$

wobei der Fehlerterm im resultierenden Geheimtext unabhängig ist vom Rauschen in den ursprünglichen Geheimtexten $\mathbf{c}_1$ und $\mathbf{c}_2$.

Durch die wiederholte Ausführung dieses Prozesses entsteht ein leveled voll homomorphes Verschlüsselungssystem. Der Schlüssel für die homomorphe Auswertung des Systems besteht aus einer Folge öffentlicher Schlüssel $(pk_1, \ldots, pk_{l+1})$ und einer Kette verschlüsselter geheimer Schlüssel $\overline{sk_1}, \ldots, \overline{sk_l}$, wobei jedes sk_i unter pk_{i+1} verschlüsselt wird. Zur homomorphen Auswertung einer Funktion f wird diese als Schaltkreis ausgedrückt, dessen Gatter topologisch in Ebenen angeordnet sind. Für die Auswertung eines Gatters in Ebene $i+1$ verwendet das System als Input den verschlüsselten geheimen Schlüssel $\overline{sk_i}$ und die unter sk_i verschlüsselten Geheimtexte aus den anliegenden Gattern der Ebene i. Diese werden wie beschrieben ausgewertet zu einem unter pk_{i+1} verschlüsselten Geheimtext, welcher einem anliegenden Gatter in Ebene $i+2$ als Input dient.

Gentrys Bootstrapping Theorem unterscheidet zwei Varianten. Wie gezeigt lässt sich für jedes System, welches Schaltkreise genügend großer Tiefe auswerten kann, ohne weitere Annahmen *leveled fully homomorphic encryption* erreichen.

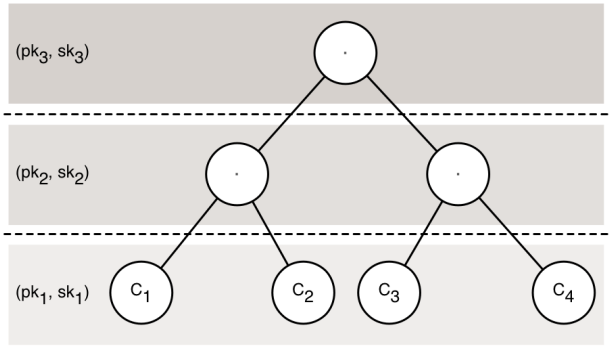

Abbildung 3.1: Bootstrapping [Gen09]. Nach jeder Auswertung eines Gatters werden die unter pk_i verschlüsselten Geheimtexte durch den beschriebenen Recrypt-Vorgang mit reduziertem Rauschen unter einem neuen Schlüssel pk_{i+1} verschlüsselt. Die dazu benötigte Folge von öffentlichen Schlüsseln pk_i und verschlüsselten geheimen Schlüsseln $\mathsf{Enc}_{\mathcal{E}}(pk_{i+1}, sk_i)$ für $0 < i \leq L$ bilden den Schlüssel für die homomorphe Auswertung *evk*.

Theorem 3.1 (Bootstrapping Theorem [Gen10])**.** *Das Verschlüsselungssystem $\mathcal{E}$ sei bootstrappable, dann existiert ein leveled fully homomorphic encryption scheme wie in Definition 3.8.*

Dabei hängt die Komplexität von $\mathsf{Gen}_{\mathcal{E}}(\lambda, L)$ sowie die Länge der erzeugten Schlüssel von der Tiefe des auszuwertenden Schaltkreises L ab, alle anderen Parameter des Systems sind unabhängig von L.

Unter der zusätzlichen Annahme, dass das zu Grunde liegende Verschlüsselungssystem auch *circular secure* ist, dass es also sicher ist, den geheimen Schlüssel des Systems mit dem eigenen öffentlichem Schlüssel zu verschlüsseln, lässt sich das System vereinfachen und ein voll homomorphes Verschlüsselungssystem erzeugen. Der Schlüssel für die homomorphe Auswertung besteht in diesem Fall lediglich aus einem Paar aus öffentlichen Schlüssel *pk* sowie einer Verschlüsselung des geheimen Schlüssels *sk* unter *pk*, welches allen Ebenen des auszuwertenden Schaltkreises zugeordnet wird.

Theorem 3.2 (Bootstrapping under circular security assumption [Gen10])**.** *Das Verschlüsselungssystem $\mathcal{E}$ sei* bootstrappable *und zusätzlich* circular secure *dann existiert ein voll homomorphes Verschlüsselungssystem wie in Definition 3.7.*

3.3.3 Squashing

Der dritte Schritt in Gentrys Schema stellt sicher, dass das gewählte SWHE Schema Bootstrapping zulässt. Da aktuell kein somewhat homomorphes Verschlüsselungssy-

stem in der Lage ist, die eigene Entschlüsselungsfunktion homomorph auszuwerten, besteht dieser Schritt darin, das gewählte Schema so zu verändern, dass die homomorphen Eigenschaften erhalten bleiben, aber die Entschlüsselung weit genug vereinfacht wird. Gentry fügte dazu dem öffentlichen Schlüssel einen verschlüsselten Hinweis auf den geheimen Schlüssel des Systems hinzu und konnte so die Komplexität von $\mathsf{Dec}_{\mathcal{E}}$ soweit reduzieren, dass sein neues Schema *bootstrappable* wurde. Die Sicherheit des neuen Systems basiert dabei auf einer neuen Annahme über die Schwere des sparse subset sum problem.

Kapitel 4

Das Approximate Eigenvector Schema

Als ein konkretes Beispiel eines aktuellen homomorphen Verschlüsselungssystems wird im folgenden Kapitel das für die Implementierung ausgewählte LWE basierte *Approximate Eigenvektor Schema* von Gentry, Sahai und Waters (GSW) [GSW13] analysiert. Während der öffentliche Schlüssel des Systems identisch ist zu Regevs ursprünglichem LWE Schema [Reg05], nutzt das System eine neue Darstellung der Geheimtexte. Statt Geheimtexte als Vektoren abzubilden wie in Kapitel 2.5.1 beschrieben, bildet das GSW Schema verschlüsselte Nachrichten auf Matrizen ab. Auf diese Art lassen sich homomorphe Operationen als Matrixmultiplikation und Addition umsetzen, was die Komplexität der Operationen im Vergleich zu anderen aktuellen Systemen reduziert. Das Format der Geheimtexte wird durch die homomorphe Addition und Multiplikation nicht verändert, es ist demnach kein zusätzlicher Schritt nötig, um nach einer homomorphen Operation die Geheimtexte wieder in ihr ursprüngliches Format zu überführen wie in anderen LWE basierten Systemen [Bra12, BV11a, BV11b]. Ein zusätzlicher Vorteil ist, dass homomorphe Berechnungen ohne einen zusätzlichen nutzer-spezifischen Schlüssel *evk* durchgeführt werden können. Diese Tatsache ermöglicht neue kryptographische Konstruktionen wie *identity-based FHE* oder *attribute-based FHE* [GSW13].

Die dem Verfahren zugrunde liegende Idee ist, dass der geheime Schlüssel $\mathbf{v}$ aus einem Eigenvektor (siehe Kapitel 2.3.2) der codierten Matrix $\mathbf{C}$ gebildet wird, auf dessen Komponenten ein betragskleiner zufälliger Fehler addiert wurde. Gentry, Sahai und Waters sprechen von einem *Approximate Eigenvector*. Die von $\mathbf{C}$ verschlüsselte Nachricht μ ist der *Eigenwert* zu $\mathbf{C}$ und dem verwendeten Eigenvektor.

Das Schema wird verständlich, wenn man sich eine fehlerfreie Version des Systems vorstellt, in der der geheime Schlüssel einen exakten Eigenvektor der verschlüssel-

ten Matrizen darstellt. Hier würde für einen Geheimtext $\mathbf{C}_i = \mathsf{Enc}_{\mathcal{E}}(\mu_i)$ gelten, dass $\mathbf{C}_i \cdot \mathbf{v} = \mu_i \cdot \mathbf{v}$. Haben zwei Matrizen $\mathbf{C}_1$ und $\mathbf{C}_2$ einen gemeinsamen Eigenvektor $\mathbf{v}$ mit den Eigenwerten μ_1 und μ_2, dann ist $\mathbf{v}$ ebenfalls Eigenvektor des Produkts $\mathbf{C}_1 \cdot \mathbf{C}_2$ und der Summe $\mathbf{C}_1 + \mathbf{C}_2$ der beiden Matrizen mit Eigenwert $\mu_1 \cdot \mu_2$ beziehungsweise $\mu_1 + \mu_2$, das Schema ist also homomorph bezüglich der Addition und der Multiplikation der Geheimtexte.

Da die Sicherheit des Schemas auf dem LWE Problem basiert, kann es sich beim geheimen Schlüssel $\mathbf{v}$ nicht um einen genauen Eigenvektor handeln, es muss vielmehr gelten, dass $\mathbf{C}$ die Nachricht μ verschlüsselt, wenn $C \cdot \mathbf{v} = \mu \cdot \mathbf{v} + \mathbf{e}$, wobei $\mathbf{e}$ einen betragskleinen Fehler-Vektor beschreibt. Um zu verhindern, dass der Fehler durch die homomorphen Operationen zu stark wächst, müssen bestimmte Terme innerhalb des Schemas *klein*, also deutlich kleiner als der Modulus q des Restklassenrings $\mathbb{Z}_q$, gewählt werden.

4.1 Ciphertext Flattening

Das Approximate Eigenvektor Schema nutzt verschiedene Transformationen an Vektoren aus [BV11a, BGV12, Bra12], die die Norm eines Vektors reduzieren ohne bestimmte Skalarprodukte zu beeinflussen, die Autoren sprechen von *Ciphertext Flattening*. Konkret wird so sichergestellt, dass die entstehenden Geheimtexte nur aus Einträgen aus $\{0,1\}$ bestehen, wodurch bessere Schranken für das Fehlerwachstum während der homomorphen Evaluation erreicht werden (siehe Kapitel 4.5.2). Das Skalarprodukt mit dem geheimen Schlüssel $\mathbf{v}$ und damit die korrekte Entschlüsselung bleiben unverändert.

Die Funktion $\mathsf{BitDecomp}_q(\mathbf{x})$ erzeugt eine binäre Repräsentation eines Vektors, indem für jedes Element des Vektors dessen Binärdarstellung in der Form $\{0,1\}^{\lceil \log q \rceil}$ erzeugt wird und diese konkateniert werden:

Funktion $\mathrm{BitDecomp}_q(\mathbf{x})$

Eingabe : $\mathbf{x} \in \mathbb{Z}_q^n$

Ausgabe : $\mathbf{x}' = (w_{1,\lceil \log q \rceil - 1}, \ldots, w_{1,0}, \ldots, w_{n,\lceil \log q \rceil - 1}, \ldots, w_{n,0}) \in \{0,1\}^{n \cdot \lceil \log q \rceil}$ mit $w_{i,j} \in \{0,1\}$ und $x[i] = \sum_{j=0}^{\lceil \log q \rceil - 1} 2^j \cdot w_{i,j} \mod q$

Die zu $\mathsf{BitDecomp}_q$ inverse Funktion $\mathsf{BitDecomp}_q^{-1}(\mathbf{x}')$ berechnet aus einem mit $\mathsf{BitDecomp}_q(\mathbf{x})$ erzeugten Vektor $\mathbf{x}'$ den ursprünglichen Vektor $\mathbf{x}$. Die Funktion ist nicht nur für binäre $\{0,1\}^{n \cdot \lceil \log q \rceil}$-Vektoren als Eingangswerte definiert, sondern für beliebige Vektoren $\mathbf{x} \in \mathbb{Z}_q^{n \cdot \lceil \log q \rceil}$.

Funktion $\text{BitDecomp}_q^{-1}(\mathbf{x})$

Eingabe : $\mathbf{x}' = (x_{1,\lceil \log q \rceil -1}, \ldots, x_{1,0}, \ldots, x_{n,\lceil \log q \rceil -1}, \ldots, x_{n,0}) \in \mathbb{Z}_q^{n \cdot \lceil \log q \rceil}$
Ausgabe : $\mathbf{x} = (\sum_{j=0}^{\lceil \log q \rceil -1} 2^j \cdot x_{1,j}, \ldots, \sum_{j=0}^{\lceil \log q \rceil -1} 2^j \cdot x_{n,j}) \in \mathbb{Z}_q^n$

Die Funktion $\text{PowersOfTwo}_q(\mathbf{x})$ multipliziert einen Vektor x komponentenweise mit der Folge $(2^{\lceil \log q \rceil}, \ldots, 2^0)$:

Funktion $\text{PowersOfTwo}_q(\mathbf{x})$

Eingabe : $\mathbf{x} \in \mathbb{Z}^n$
Ausgabe : $[(2^{\lceil \log q \rceil -1} x[0], \ldots, 2^0 \cdot x[0], \ldots, 2^{\lceil \log q \rceil -1} \cdot x[n], \ldots, 2^0 \cdot x[n])]_q \in \mathbb{Z}_q^{n \cdot \lceil \log q \rceil}$

Die Funktion $\text{Flatten}_q(\mathbf{x})$ erzeugt aus einem Vektor der Länge $n \cdot \lceil \log q \rceil$ über $\mathbb{Z}_q$ einen ebenfalls $n \cdot \lceil \log q \rceil$-dimensionalen Vektor mit Komponenten aus $\{0, 1\}$.

Funktion $\text{Flatten}_q(\mathbf{x})$

Eingabe : $\mathbf{x} \in \mathbb{Z}^{n \cdot \lceil \log q \rceil}$
Ausgabe : $\text{BitDecomp}_q(\text{BitDecomp}_q^{-1}(\mathbf{x})) \in \{0, 1\}^{n \cdot \lceil \log q \rceil}$

Für Matrizen seien die beschriebenen Funktionen definiert als die Anwendung der Funktion auf jeden Zeilenvektor der Matrix.

Die beschriebenen Transformationen haben folgende nützliche Eigenschaften [GSW13]. Das Skalarprodukt aus einem mittels $\text{BitDecomp}_q(\mathbf{x})$ in seine Bitdarstellung zerlegten Vektor $\mathbf{x}$ und dem Ergebnis von $\text{PowersOfTwo}_q(\mathbf{y})$ ist identisch zum Skalarprodukt der unveränderten Vektoren $\mathbf{x}$ und $\mathbf{y}$.

$$\langle \mathbf{x}, \mathbf{y} \rangle = \langle \text{BitDecomp}_q(\mathbf{x}), \text{PowersOfTwo}_q(\mathbf{y}) \rangle \text{ für alle } \mathbf{x}, \mathbf{y} \in \mathbb{Z}_q^n \tag{4.1}$$

Die Funktion Flatten_q nutzt diese Tatsache. Sie reduziert die Elemente eines beliebigen Vektors aus $\mathbb{Z}_q^{n \cdot \lceil \log q \rceil}$ auf Werte aus $\{0, 1\}$, ohne dessen Skalarprodukt mit $\text{PowersOfTwo}_q(y)$ zu verändern. Für alle $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_q^n$ und $\mathbf{x}' \in \mathbb{Z}_q^{n \cdot \lceil \log q \rceil}$ gilt

$$\begin{aligned} \langle x', \text{PowersOfTwo}_q(y) \rangle &= \langle \text{BitDecomp}_q^{-1}(x'), y \rangle \\ &= \langle \text{Flatten}_q(x'), \text{PowersOfTwo}_q(y) \rangle. \end{aligned} \tag{4.2}$$

4.2 Verschlüsselung und Reduktion auf LWE

Das dem Schema zugrunde liegende Verschlüsselungssystem basiert auf dem Learning with Errors Problem (siehe Kapitel 2.5.1). Im ersten Schritt wird eine LWE-

Instanz aufgesetzt, deren Parameter so gewählt werden müssen, dass das beschriebene LWE Problem die gewünschte Sicherheit gegen bekannte Angriffe gewährleistet. Weiterhin müssen die Parameter so gewählt werden, dass die homomorphe Auswertung von Schaltkreisen der gewünschten Tiefe L möglich ist.

Setup Konkret wird abhängig vom gewünschten Sicherheitsparameter λ und der gewünschten Tiefe L der Modulus q des Restklassenrings $\mathbb{Z}_q$, die Dimension des Gitters n sowie die Wahrscheinlichkeitsverteilung χ für den Fehler gewählt. Zusätzlich wird der Parameter $m = O(n \log q)$ festgelegt, welcher angibt, aus wie vielen LWE Samples eine Verschlüsselung besteht. Die Dimension der quadratischen Geheimtext-Matrizen N sei $(n+1) \cdot \lceil \log q \rceil$.

Schlüsselerzeugung Für die Schlüssel des Systems wird folgende Darstellung gewählt [GSW13]: Als Geheimnis des Systems dient ein n-dimensionaler Vektor $s \in \mathbb{Z}_q^n$. Der öffentliche Schlüssel pk besteht aus m LWE Samples $(\langle \mathbf{a}_i, \mathbf{s} \rangle + e_i, \mathbf{a}_i) \in \mathbb{Z}_q^{n+1}$ (siehe Definition 2.10), die als Reihen einer Matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times (n+1)}$ interpretiert werden. Das ursprüngliche Geheimnis $\mathbf{s}$ wird umgeformt zu $\mathbf{s} = (1, -\mathbf{s})$, so dass $\mathbf{A} \cdot \mathbf{s} = \mathbf{e}$ mit $\mathbf{e} \sim \chi^m$ gilt. Die Matrix $\mathbf{A}$ bildet den öffentlichen Schlüssel des Systems, als geheimer Schlüssel wird der von Gentry, Sahai und Waters *approximate Eigenvektor* genannte Vektor $\mathbf{v} = \mathsf{PowersOfTwo}_q(\mathbf{s}) \in \mathbb{Z}_q^N$ verwendet.

Ver- und Entschlüsselung Um ein Bit $\mu \in \{0,1\}$ zu verschlüsseln, wird eine zufällige Matrix $\mathbf{R} \in \{0,1\}^{N \times m}$ erzeugt und der Geheimtext $\mathbf{C} = \mathsf{Flatten}_q(\mu \cdot \mathbf{I}_N + \mathsf{BitDecomp}_q(\mathbf{R} \cdot \mathbf{A}))$ ausgegeben ($\mathbf{I}_N$ sei die N-dimensionale Identitätsmatrix). Da durch die Anwendung der Funktion $\mathsf{Flatten}_q$ das Matrizenprodukt von $\mathbf{C}$ mit dem geheimen Schlüssel $\mathbf{v}$ nicht verändert wird, gilt:

$$\begin{aligned} \mathbf{C} \cdot \mathbf{v} &= \mu \cdot \mathbf{v} + \mathsf{BitDecomp}_q(\mathbf{R} \cdot \mathbf{A}) \cdot \mathbf{v} \\ &= \mu \cdot \mathbf{v} + \mathbf{R} \cdot \mathbf{A} \cdot \mathbf{s} \\ &= \mu \cdot \mathbf{v} + \underbrace{\mathbf{R} \cdot \mathbf{e}}_{\|\mathbf{R} \cdot \mathbf{e}\|_\infty \leq m \cdot \|\mathbf{e}\|_\infty} \end{aligned} \tag{4.3}$$

Für die Zeilenvektoren $\mathbf{c}_i$ der Matrix $\mathbf{C}$ gilt

$$\langle \mathbf{c}_i, \mathbf{v} \rangle = \mu \cdot \mathbf{v}[i] + \underbrace{\langle \mathbf{r}_i, \mathbf{e} \rangle}_{\leq \|\mathbf{e}\|_1} \tag{4.4}$$

wobei $\mathbf{v}[i]$ die i-te Komponente von $\mathbf{v}$ und $\mathbf{r}_i$ die i-te Zeile von $\mathbf{R}$ darstellen.

Die Entschlüsselung des Bits μ anhand von Gleichung 4.4 funktioniert, wenn das Rauschen im Geheimtext klein genug ist und wenn $\mathbf{v}$ ein Element $\mathbf{v}[i]$ hat, dessen Betrag groß genug ist, um $\mu \in \{1, 0\}$ zu entschlüsseln. Konkret muss der Betrag des Fehlerterms $< q/8$ sein und $q/4 \leq \mathbf{v}[i] < q/2$ gelten (siehe Kapitel 4.5).

4.3 Homomorphe Operationen

$\mathbf{C}_1$ und $\mathbf{C}_2$ seien Verschlüsselungen von μ_1 und μ_2, so daß $\mathbf{C}_i \cdot \mathbf{v} = \mu_i \cdot \mathbf{v} + \mathbf{e}_i$ für kleine $\mathbf{e}_i$.

4.3.1 Addition

Die homomorphe Addition zweier Geheimtexte entspricht der Addition der beiden Geheimtextmatrizen.

$$\mathbf{C}^+ = \mathbf{C}_1 + \mathbf{C}_2$$

Es lässt sich leicht zeigen, dass C^+ eine korrekte homomorphe Operation ist, da

$$\begin{aligned} \mathbf{C}^+ \cdot \mathbf{v} &= (\mathbf{C}_1 + \mathbf{C}_2) \cdot \mathbf{v} \\ &= \mathbf{C}_1 \cdot \mathbf{v} + \mathbf{C}_2 \cdot \mathbf{v} \\ &= (\mu_1 \cdot \mathbf{v} + \mathbf{e}_1) + (\mu_2 \cdot \mathbf{v} + \mathbf{e}_2) \\ &= (\mu_1 + \mu_2) \cdot \mathbf{v} + (\mathbf{e}_1 + \mathbf{e}_2) \end{aligned} \tag{4.5}$$

Der Fehler wächst dabei leicht wie in jedem homomorphen Verschlüsselungssystem. Solange er im Vergleich zu den codierten Nachrichten klein bleibt, lässt sich die Summe der Geheimtexte zur Summe der Nachrichten entschlüsseln. Während in dieser Arbeit grundsätzlich von bitweise verschlüsselten Nachrichten ausgegangen wird, also von Nachrichten aus dem Nachrichtenraum $\mathcal{M} = \{0, 1\}$, sei erwähnt, dass $\mathbf{C}^+$ für Nachrichten aus dem kompletten Restklassenring $\mathbb{Z}_q$ definiert ist.

4.3.2 Multiplikation

Die homomorphe Multiplikation von Geheimtexten entspricht der Multiplikation der beiden Geheimtextmatrizen.

$$\mathbf{C}^\times = \mathbf{C}_1 \cdot \mathbf{C}_2$$

Auch hier handelt es sich um eine korrekte homomorphe Operation, da

$$\begin{aligned}\mathbf{C}^{\times} \cdot \mathbf{v} &= \mathbf{C}_1 \cdot \mathbf{C}_2 \cdot \mathbf{v} \\ &= \mathbf{C}_1 \cdot (\mu_2 \cdot \mathbf{v} + \mathbf{e}_2) \\ &= \mu_2 \cdot (\mu_1 \cdot \mathbf{v} + \mathbf{e}_1) + \mathbf{C}_1 \cdot \mathbf{e}_2 \\ &= \mu_1 \cdot \mu_2 \cdot \mathbf{v} + \mu_2 \cdot \mathbf{e}_1 + \mathbf{C}_1 \cdot \mathbf{e}_2\end{aligned} \tag{4.6}$$

So lange der resultierende Fehlervektor $\mu_2 \cdot \mathbf{e}_1 + \mathbf{C}_1 \cdot \mathbf{e}_2$ klein genug ist, lässt sich das Produkt der Geheimtexte zum Produkt der Nachrichten entschlüsseln. Eine genaue Betrachtung der Fehlerwachstums unter homomorpher Berechnung findet sich in Kapitel 4.5 dieser Arbeit. Interessanterweise ist auch $\mathbf{C}_2 \cdot \mathbf{C}_1$ eine Verschlüsselung von $\mu_1 \cdot \mu_2$ obwohl Matrixmultiplikation nicht kommutativ ist. Auch $\mathbf{C}^{\times}$ lässt sich auf Verschlüsselungen von Nachrichten aus dem kompletten Restklassenring $\mathbb{Z}_q$ anwenden.

Da sich die Multiplikation zweier Werte aus $\{0, 1\}$ auch als Auswertung eines binären AND Operators interpretieren lässt, lässt sich aus $\mathbf{C}^{\times}$ direkt eine Funktion für die Auswertung der binären NAND Operation konstruieren. Im vorgestellten Schema stellt die N-dimensionale Identitätsmatrix $\mathbf{I}_N$ eine fehlerfreie Verschlüsselung von 1 dar, die $\mathsf{NOT}(\mathbf{C})$-Funktion entspricht also $\mathbf{I}_N - \mathbf{C}$. Damit wird die NAND-Funktion definiert als

$$\mathsf{NAND}(\mathbf{C}_1, \mathbf{C}_2) = \mathbf{I}_N - \mathbf{C}_1 \cdot \mathbf{C}_2.$$

4.4 Das Schema

Das implementierte Approximate Eigenvector Schema lässt sich zusammenfassend darstellen als Tupel der folgenden Algorithmen:

Setup$_{\mathcal{E}}(\lambda, L)$**:** Wähle einen Modul q von $k = k(\lambda, L)$ bits, die Dimension des Gitters $n = n(\lambda, L)$, eine Fehlerverteilung $\chi = \chi(\lambda, L)$ und einen Parameter $m = m(\lambda, L)$ passend für LWE mit einer Sicherheit von $\geq 2^{\lambda}$ gegen bekannte Angriffe. Setze $l = \lceil \log q \rceil$ und $N = (n+1) \cdot l$. Sei $params = (n, q, \chi, m)$.

SecretKeyGen$_{\mathcal{E}}(params)$**:** Wähle zufällig gleichverteilt einen Vektor $\mathbf{t} \leftarrow \mathbb{Z}_q^n$. Setze $\mathbf{s} = (1, -t_1, \ldots, -t_n) \in \mathbb{Z}_q^{n+1}$ und gib $\mathbf{v} = \mathsf{PowersOfTwo}_q(\mathbf{s})$ als Secret Key aus.

PublicKeyGen$_{\mathcal{E}}(params, \mathbf{t})$**:** Wähle zufällig gleichverteilt eine Matrix $\mathbf{B} \leftarrow \mathbb{Z}_q^{m \times n}$ und zufällig χ-verteilt einen Vektor $\mathbf{e} \leftarrow \mathcal{X}^m$. Sei $\mathbf{b} = \mathbf{B} \cdot \mathbf{t} + \mathbf{e}$ und $\mathbf{A}$ eine

$m \times (n+1)$-Matrix bestehend aus $\mathbf{b}$ gefolgt von den n Spalten von $\mathbf{B}$. Gib $\mathbf{A}$ als Public Key aus.

$\mathsf{Enc}_{\mathcal{E}}(params, \mathbf{A}, \mu)$ Um eine Nachricht $\mu \in \{0,1\}$ zu verschlüsseln, wähle zufällig gleichverteilt eine Matrix $\mathbf{R} \in \{0,1\}^{N \times m}$ und gib den Geheimtext $\mathbf{C} = \mathsf{Flatten}_q(\mu \cdot \mathbf{I}_N + \mathsf{BitDecomp}_q(\mathbf{R} \cdot \mathbf{A})) \in \mathbb{Z}_q^{N \times N}$ aus.

$\mathsf{Dec}_{\mathcal{E}}(params, \mathbf{v}, \mathbf{C})$: Sei $\mathbf{c}$ die zweite Zeile von $\mathbf{C}$. Gib $\mu = 0$ aus wenn $|\lfloor \langle \mathbf{c}, \mathbf{v} \rangle / \mathbf{v}[2] \rceil_q| \leq q/8$, sonst $\mu = 1$.

$\mathsf{MultConst}_{\mathcal{E}}(\mathbf{C}, a)$: Um einen Geheimtext $\mathbf{C} \in \mathbb{Z}_q^{N \times N}$ mit einer Konstanten $a \in \mathbb{Z}_q$ zu multiplizieren, gib $\mathsf{Flatten}_q(\mathsf{Flatten}_q(a \cdot \mathbf{I}_N) \cdot \mathbf{C})$ aus.

$\mathsf{Add}_{\mathcal{E}}(\mathbf{C}_1, \mathbf{C}_2)$: Gib $\mathsf{Flatten}_q(\mathbf{C}_1 + \mathbf{C}_2)$ aus.

$\mathsf{Mult}_{\mathcal{E}}(\mathbf{C}_1, \mathbf{C}_2)$: Gib $\mathsf{Flatten}_q(\mathbf{C}_1 \cdot \mathbf{C}_2)$ aus.

$\mathsf{NAND}_{\mathcal{E}}(\mathbf{C}_1, \mathbf{C}_2)$: Gib $\mathsf{Flatten}_q(\mathbf{I}_N - \mathbf{C}_1 \cdot \mathbf{C}_2)$ aus.

Die Funktionen $\mathsf{BitDecomp}_q$, $\mathsf{BitDecomp}_q^{-1}$, $\mathsf{PowersOfTwo}_q$ und $\mathsf{Flatten}_q$ seien definiert wie in Kapitel 4.1 beschrieben.

4.5 Korrektheit

Um sicherzustellen, dass das Schema beliebige Nachrichten korrekt ver- und entschlüsselt, und dass die homomorphe Auswertung von Funktionen der gewünschten Komplexität in jedem Fall korrekte Ergebnisse liefert, müssen bestimmte Einschränkungen für die gewählten Systemparameter q, n und χ gelten. Diese werden im Folgenden für die korrekte Entschlüsselung und die homomorphe Evaluation untersucht.

4.5.1 Ver- und Entschlüsselung

Für einen Geheimtext $\mathbf{C} \leftarrow \mathsf{Enc}_{\mathcal{E}}(pk, \mu)$ gilt nach der Definition des Schemas

$$\mathbf{C} \cdot \mathbf{v} = \mu \cdot \mathbf{v} + \mathbf{e}$$

wobei $\mathbf{e}$ einen betragskleinen Fehlervektor darstellt. Damit lässt sich der Fehler innerhalb eines Geheimtexts folgendermassen definieren:

Definition 4.1 (Rauschen innerhalb eines Geheimtextes [BV14]). *Für jeden Geheimtext* $\mathbf{C} \in \{0,1\}^{N\times N}, \mathbf{v} \leftarrow \mathsf{SecretKeyGen}(q,n)$ *und* $\mu \in \mathbb{Z}_q$ *sei*

$$noise_{\mathbf{v},\mu}(\mathbf{C}) := \|(\mathbf{C} - \mu \cdot \mathbf{I}_N) \cdot \mathbf{v}\|_\infty$$

$noise_{\mathbf{v},\mu}(\mathbf{C})$ bezeichnet die Maximumsnorm des Fehlervektors $\mathbf{e}$, also den grössten Betrag einer Komponente von $\mathbf{e}$. Mit dieser Definition lässt sich zeigen, dass jeder Geheimtext, dessen Rauschen kleiner ist als $q/8$, korrekt entschlüsselt werden kann.

Behauptung 1. *Für jeden Geheimtext* $\mathbf{C} \leftarrow \mathsf{Enc}_{\mathcal{E}}() \in \{0,1\}^{N\times N}$, *jeden Schlüssel* $\mathbf{v} \leftarrow \mathsf{SecretKeyGen}_{\mathcal{E}}(q,n)$ *und jedes* $\mu \in \{0,1\}$ *mit* $noise_{\mathbf{v},\mu}(\mathbf{C}) < q/8$ *gilt*

$$\mathsf{Dec}_{\mathcal{E}}(\mathbf{v},\mathbf{C}) = \mu$$

Beweis. [BV14] Es gilt, dass $noise_{\mathbf{v},\mu}(C) < q/8$, also ist

$$C \cdot \mathbf{v} = \mu \cdot \mathbf{v} + \mathbf{e} \text{ mit } \|\mathbf{e}\|_\infty < q/8.$$

Für den zweiten Zeilenvektor von C gilt dann

$$\langle \mathbf{c}_2, \mathbf{v}\rangle = \mu \cdot 2^{\lceil \log q\rceil - 2} + e_2 \text{ mit } |e_2| < q/8.$$

Damit lässt sich die korrekte Entschlüsselung der möglichen Nachrichten $\mu = 0$ und $\mu = 1$ zeigen. Für $\mu = 0$ gilt

$$|[\langle \mathbf{c}_2, \mathbf{v}\rangle]_q| = |e_2| < q/8$$

für $m = 1$ gilt

$$|[\langle \mathbf{c}_2, \mathbf{v}\rangle]_q| = 2^{\lceil \log q\rceil - 2} + e_2 \geq q/4 - |e_2| \geq q/8$$

da $q/4 \leq 2^{\lceil \log q\rceil - 2} < q/2$.

Aus der Definition von $\mathsf{Dec}_{\mathcal{E}}(\mathbf{v},\mathbf{C})$ folgt die korrekte Entschlüsselung für beliebige Nachrichten $\mu \in 0, 1$. □

4.5.2 Homomorphe Evaluation

Wie in allen homomorphen Verschlüsselungssystemen wächst der in Geheimtexten enthaltene Fehler während homomorpher Berechnungen. Das bisher entwickelte System kann wenige aufeinander folgende Operationen korrekt homomorph auswerten, es handelt sich also um ein *Somewhat Homomorphic Encryption Scheme*. Werden die auszuwertenden Funktionen allerdings komplexer, wird der Fehler in

den Ergebnissen schnell so groß dass eine korrekte Entschlüsselung nicht mehr gewährleistet ist.

Für die Entwicklung eines *Leveled Homomorphic Encryption System*, welches Schaltkreise gegebener Tiefe auswerten kann, wird zunächst der Einfluss der einzelnen homomorphen Operationen auf das Fehlerwachstum analysiert. Ziel ist es, Kriterien für die Systemparameter festzulegen, die die Auswertung von Schaltkreisen einer gewünschten Tiefe ermöglichen, ohne die Sicherheit des Systems zu reduzieren.

Addition

Werden zwei Geheimtexte in einem GSW System addiert, ist das im Ergebnis enthaltene Rauschen (siehe Definition 4.1) höchstens so gross wie die Summe des Rauschens der beiden Eingangswerte.

Behauptung 2 (Fehlerwachstum der homomorphen Addition [BV14])**.** *Für jedes* $\mathbf{v} \leftarrow \mathsf{SecretKeyGen}(q, n)$, $\mathbf{C}_1 \leftarrow \mathsf{Enc}_{\mathcal{E}}(pk, \mu_1)$, $\mathbf{C}_2 \leftarrow \mathsf{Enc}_{\mathcal{E}}(pk, \mu_2)$ *gilt*

$$noise_{\mathbf{v},\mu_1+\mu_2}(\mathsf{Add}(\mathbf{C}_1, \mathbf{C}_2)) \leq noise_{\mathbf{v},\mu_1}(\mathbf{C}_1) + noise_{\mathbf{v},\mu_2}(\mathbf{C}_2)$$

Beweis. Sei $\mathbf{C}^+ \leftarrow \mathsf{Add}(\mathbf{C}_1, \mathbf{C}_2)$.

$$\begin{aligned} \mathbf{C}^+ \cdot \mathbf{v} &= \mathsf{Flatten}_q(\mathbf{C}_1 + \mathbf{C}_2) \cdot \mathbf{v} \\ &= (\mathbf{C}_1 + \mathbf{C}_2) \cdot \mathbf{v} \\ &= \mathbf{C}_1 \cdot \mathbf{v} + \mathbf{C}_2 \cdot \mathbf{v} \\ &= (\mu_1 \cdot \mathbf{v} + \mathbf{e}_1) + (\mu_2 \cdot \mathbf{v} + \mathbf{e}_2) \\ &= (\mu_1 + \mu_2) \cdot \mathbf{v} + \mathbf{e}_1 + \mathbf{e}_2 \end{aligned} \tag{4.7}$$

Damit folgt aus Definition 4.1 $noise_{\mathbf{v},\mu_1+\mu_2}(\mathbf{C}^+) \leq noise_{\mathbf{v},\mu_1}(\mathbf{C}_1) + noise_{\mathbf{v},\mu_2}(\mathbf{C}_2)$. □

Multiplikation

Wie in Gleichung 4.6 gezeigt wurde, gilt für das Ergebnis einer Multiplikation zweier Geheimtexte

$$\mathbf{C}_1 \cdot \mathbf{C}_2 \cdot \mathbf{v} = \mu_1 \cdot \mu_2 \cdot \mathbf{v} + \underbrace{\mu_2 \cdot \mathbf{e}_1}_{\leq \mu_2 \cdot |\mathbf{e}_1|_\infty} + \underbrace{\mathbf{C}_1 \cdot \mathbf{e}_2}_{\leq N \cdot |\mathbf{e}_2|_\infty}$$

Der Betrag des Fehlers in $\mathbf{C}^\times$ hängt also nicht nur vom Rauschen der Eingangsvariablen ab, sondern zusätzlich vom Betrag der Nachricht μ_2 und der Norm des Geheimtexts $\mathbf{C}_1$. Um das Fehlerwachstum möglichst gering zu halten, werden Geheimtexte mit Hilfe von $\mathsf{Flatten}_q$ in Matrizen aus $\{0,1\}^{N \times N}$ überführt und der

Nachrichtenraum auf $\{0,1\}$ beschränkt, so dass für das Fehlerwachstum während einer homomorphen Multiplikation gilt:

$$\begin{aligned} noise_{\mathbf{v},\mu_1\cdot\mu_2}(\mathbf{C}^{\times}) &\leq noise_{\mathbf{v},\mu_1}(\mathbf{C}_1) + N \cdot noise_{\mathbf{v},\mu_2}(\mathbf{C}_2) \\ &\leq (N+1) \cdot max\{noise_{\mathbf{v},\mu_1}(\mathbf{C}_1), noise_{\mathbf{v},\mu_2}(\mathbf{C}_2)\} \end{aligned} \tag{4.8}$$

Das Gleiche gilt für das Fehlerwachstum während der homomorphen Auswertung der NAND-Funktion [GSW13].

$$\begin{aligned} \mathsf{NAND}(\mathbf{C}_1,\mathbf{C}_2)\cdot\mathbf{v} &= (\mathbf{I}_N - \mathbf{C}_1\cdot\mathbf{C}_2)\cdot\mathbf{v} \\ &= (1-\mu_1\cdot\mu_2)\cdot\mathbf{v} + \underbrace{\mu_2\cdot\mathbf{e}_1}_{\leq\mu_2\cdot|\mathbf{e}_1|_\infty} + \underbrace{\mathbf{C}_1\cdot\mathbf{e}_2}_{\leq N\cdot|\mathbf{e}_2|_\infty} \end{aligned} \tag{4.9}$$

Es fällt auf, dass der Fehler innerhalb der Eingangsvariablen asymmetrisch in das Ergebnis eingeht. Während $noise_{\mathbf{v},\mu_1}(C_1)$ mit null oder eins multipliziert wird, geht $noise_{\mathbf{v},\mu_2}(C_2)$ maximal N-fach in das Ergebnis ein. Die Reihenfolge der Multiplikation beeinflusst demnach das Fehlerwachstum, wie in Abbildung 4.1 gezeigt wird. Die im Rahmen dieser Arbeit entstandene Implementierung des Schemas verwendet eine Idee von Brakerski und Vaikuntanathan [BV14], die diese Beobachtung nutzt, um durch einen geschickten Aufbau der auszuwertenden Schaltkreise das Fehlerwachstum einzuschränken.

Auswertung von Schaltkreisen

Durch iterative Anwendung der beschriebenen homomorphen Operationen lassen sich unterschiedliche Berechnungen auf den verschlüsselten Daten ausführen.

Im einfachsten Fall wird die homomorph auszuwertende Funktion durch einen Booleschen Schaltkreis dargestellt. Da die NAND-Verknüpfung alle logischen Verknüpfungen nachbilden kann, NAND also eine *vollständige Basis* logischer Operatoren darstellt, lässt sich jeder binäre Schaltkreis so umformen, dass er komplett aus NAND-Gattern aufgebaut ist. Der Fehler des verschlüsselten Ergebnisses wird dann nur durch die homomorphe Auswertung von NAND-Operationen beeinflusst.

Um einen Geheimtext korrekt zu entschlüsseln, darf der Betrag des Fehlervektors $q/8$ nicht übersteigen. Wie in Gleichung 4.8 gezeigt wurde, erhöht sich der maximale Fehler $noise_{\mathbf{v},in}$ eines Geheimtextes in einer NAND-basierten Schaltung pro Gatter höchstens um den Faktor $N+1$. Um für eine NAND-Schaltung der Tiefe L und

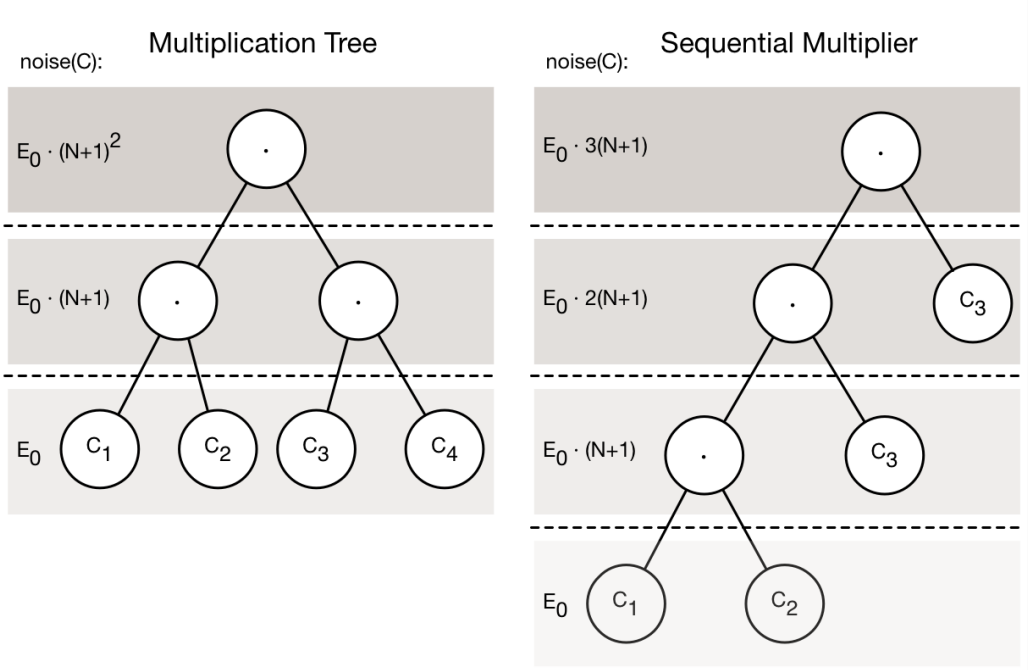

Abbildung 4.1: Sequentielle Auswertung[BV11a]. Die intuitiven Annahme, dass das Rauschen innerhalb der Geheimtexte nach der Auswertung eines Schaltkreises nur von dessen Tiefe bestimmt wird, gilt im GSW System nicht. Werden l Geheimtexte $C_1, \ldots, C_l$ mit einem Rauschen von E_0 miteinander multipliziert, wächst der Fehler in den Geheimtexten bei einer Auswertung als Multiplikationsbaum um den Faktor $(N+1)^{\log l}$, während eine sequentielle Auswertung ein Fehlerwachstum von $(l-1)(N+1)$ bewirkt.

frisch verschlüsselte Geheimtexte mit einem Fehler von $noise_{\mathbf{v},in}(\mathbf{C})$ eine korrekte Dekodierung der Ergebnisse zu garantieren, muss also gelten [GSW13]:

$$(N+1)^L \cdot noise_{\mathbf{v},in}(\mathbf{C}) \leq q/8 \tag{4.10}$$

Wählt man die Parameter des LWE Systems so, dass Gleichung 4.10 gilt, lassen sich NAND-basierte Schaltkreise der Tiefe L homomorph auswerten.

Die Analyse von Systemen, die nicht mit bitweise verschlüsselten Nachrichten operieren, sondern einen größeren Nachrichtenraum nutzen und Funktionen entsprechend als aus Additions- und Multiplikationsgattern aufgebaute arithmetische Schaltkreise auswerten, gestaltet sich schwieriger und wird im Rahmen dieser Arbeit nicht näher betrachtet. Im Fall von Multiplikationsgattern hängt der Anstieg des Fehlers unter anderem vom Wert des zweiten Eingangsparameters ab. Um das Wachstum des Fehlers trotzdem nach oben abschätzen zu können, ist es beispielsweise möglich, einen Wert T als obere Schranke für die Eingangsvariablen zu

definieren und den arithmetischen Schaltkreis so zu wählen, dass für entsprechend beschränkte Eingangsvariablen auch alle Zwischenergebnisse eine zweite obere Schranke T' nicht übersteigen. Der Fehler im verschlüsselten Endergebnis hat in diesem Fall höchstens einen Betrag von $(N + T')^L \cdot noise_{\mathbf{v},in}$, wobei L die Tiefe des auszuwertenden Schaltkreises und $noise_{\mathbf{v},in}$ den maximalen Fehlerbetrag eines frisch verschlüsselten Geheimtexts angibt [GSW13].

4.5.3 Voll homomorphe Verschlüsselung

Lässt sich zeigen, dass das System in der Lage ist, den eigenen Entschlüsselungsalgorithmus homomorph auszuwerten, kann es durch Bootstrapping in ein voll homomorphes Verschlüsselungssystem überführt werden (siehe Kapitel 3.3.2). Die Entschlüsselungsfunktion des Schemas ist im wesentlichen identisch zu Regevs ursprünglichem LWE Verschlüsselungssystem. Folgender Satz aus [BV11a] gilt auch für das Approximate Eigenvector System [BV14]:

Satz 4.1. *Es gibt eine Konstante c_{dec}, so dass $\mathsf{Dec}_{\mathcal{E}}$, als Schaltkreis beschrieben, mit den Parametern n, q höchstens eine Tiefe von $c_{dec} \cdot \log(n \log q)$ hat.*

In der bisher beschriebenen Form ist das System nicht in der Lage, Schaltkreise der erforderlichen Tiefe auszuwerten. Eine Lösung des Problems besteht darin, das asymmetrische Fehlerwachstum während einer homomorphen Multiplikation zu nutzen und komplexe Schaltkreise vor der Auswertung zu sequentialisieren. Wie bereits in Kapitel 4.5.2 gezeigt, wächst der Fehler bei einer sequentiellen Auswertung einer Multiplikation von L Geheimtexten mit gleichem Rauschen um $L \cdot poly(n)$ während die Auswertung in Form eines Baumes zu einem deutlich größeren Fehler von $poly(n)^{\log L}$ führt.

Brakerski und Vaikuntanathan [BV14] entwickelten eine Variante des GSW Systems, die hierzu Barringtons Theorem [Bar86] nutzt. Dieses wurde 1986 von David Barrington aufgestellt und besagt, dass sich beliebige boolesche Schaltkreise polynomieller Größe und logarithmischer Tiefe als *Permutation Branching Program* mit einer Breite von fünf und polynomieller Länge darstellen lassen (siehe Abbildung 2.2). Brakerski und Vaikuntanathan zeigen, wie sich ein solches Branching Programm im GSW System homomorph als Sequenz von Multiplikationen auswerten lässt [BV14]. Da die Entschlüsselungsfunktion des Schemas in der Klasse der Schaltkreise liegt, für die Barringtons Theorem gilt, lässt sich das Bootstrapping Theorem [Gen09] anwenden und ein voll homomorphes Verschlüsselungssystem konstruieren, welches beliebige Funktionen homomorph auswerten kann.

4.6 Parameterwahl

Um die Sicherheit des Approximate Eigenvector Systems und die korrekte homomorphe Evaluation zu gewährleisten, gelten verschiedene, teilweise widersprüchliche Anforderungen an die Systemparameter.

Die Sicherheit des Systems wird unter anderem bestimmt durch das Verhältnis des Dimensionsparameters n zum maximalen Betrag der Fehlerverteilung χ, welches die relative Schwere des zugrunde liegenden DLWE-Problems festlegt. Ein grösserer maximaler Fehler B im Vergleich zu n erschwert Angriffe auf das System.

Andererseits muss für eine korrekte homomorphe Auswertung eines Schaltkreises der Tiefe L das Verhältnis des Modulus q zu B exponentiell mit L wachsen. Um möglichst viele homomorphe Operationen zu ermöglichen, müsste q möglichst gross gewählt werden und der maximale Fehler B möglichst klein. Da es eine untere Grenze dafür gibt, wie klein B gewählt werden kann, muss q exponentiell mit L wachsen [BL14].

Gleichzeitig bestimmen die gewählten Systemparameter die Laufzeit der Algorithmen sowie den Speicherbedarf des Systems. Die Größe der generierten Schlüssel und Geheimtexte hängt linear von n und logarithmisch von q ab. Um ein möglichst effizient arbeitendes System zu erreichen, wäre es also wünschenswert, die Parameter so klein wie möglich zu wählen.

Konkret müssen die gewählten Parameter folgende Kriterien erfüllen:

Sicherheit:

- $n = O(\log(q/B))$ [GSW13].
- $m = O(n \log q)$ [GSW13].
- q sei polynomiell in n beschränkt [Reg05].

Korrektheit:

- $q/B > 8(n\lceil \log q \rceil + 1)^L$ [GSW13].

Da die Parameter sich gegenseitig beeinflussen, ist die Wahl eines Settings, welches die genannten Einschränkungen erfüllt, relativ komplex. Tabelle 4.1 zeigt mögliche Parameterkombinationen für ein Approximate Eigenvector Schema mit Modulus q, Dimensionsparametern n und m und einer diskreten Gaußverteilung $D_{\mathbb{Z},\alpha q}$ mit einer Standardabweichung von αq als Fehlerverteilung.

Parameter	Gentry, Sahai, Waters [GSW13]	Braskerski, Vaikuntanathan [BV14]
n	$O(\lambda)$	$O(\lambda)$
q	$2^{L \log n}$	$O(poly(n))$
α	$1/(\sqrt{n}\log^2 n)$	$1/\tilde{O}(\sqrt{n \log q} \cdot 4^L)$
m	$2n \log q$	$(n+1) \cdot (\log q + O(1))$

Tabelle 4.1: Beispielhafte Parametersettings aus [GSW13] und [BV14]

Kapitel 5

Implementierung

Teil dieser Arbeit war die praktische Umsetzung des im vorigen Kapitel vorgestellten Leveled homomorphen Verschlüsselungsverfahrens von Gentry, Sahai und Waters [GSW13]. Ziel war es, theoretische Aussagen zu Komplexität und homomorphen Eigenschaften des Systems anhand eines praktischen Beispiels zu überprüfen.

Im Rahmen der Implementierung zeigte sich, dass das ursprüngliche GSW-Schema zur homomorphen Auswertung komplexerer Schaltkreise extrem große Systemparameter benötigt, so dass insbesondere der Arbeitsspeicher des verwendeten Testrechners (2,4 GHz Intel Core i5 CPU mit 8 GB DDR3 RAM) für die Auswertung von Schaltkreisen der Tiefe $L < 2$ nicht mehr ausreichte.

Es entstand eine zweite Variante des Schemas, die verschiedene Optimierungen von Brakerski und Vaikuntanathan [BV14] nutzt und so die Auswertung tieferer Schaltkreise ermöglicht.

5.1 Verwendete Technologien

Für die prototypische Implementierung wurde das Computeralgebrasystem *Sage* [S$^+$09] eingesetzt. Sage ist pythonbasiert und stellt viele der für die Implementierung benötigten grundlegenden Funktionen aus dem Bereich der Algebra und der Zahlentheorie zur Verfügung.

5.2 Optimierungen

Die Implementierung setzt im Wesentlichen das in Kapitel 4.4 beschriebene homomorphe Schema um. Ergänzend wurde eine Möglichkeit implementiert, Integer-

Nachrichten aus dem kompletten Ring $\mathbb{Z}_q$ statt einzelner Bits in einem Geheimtext-Block zu verschlüsseln. Die entstandenen Geheimtexte lassen sich homomorph addieren und multiplizieren, es gelten allerdings die in Kapitel 4.5.2 beschriebenen Einschränkungen bezüglich des Fehlerwachstums.

Zusätzlich wurde eine Variante des Systems entwickelt, in der das Rauschen innerhalb der Geheimtexte nicht mehr exponentiell sondern linear mit der Tiefe der ausgewerteten Schaltkreise wächst.

5.2.1 $\mathbb{Z}_q$ als Nachrichtenraum

Zur Entschlüsselung beliebiger Nachrichten $\mu \in \mathbb{Z}_q$ wird ein von Micchiano und Peikert [MP12] entwickelter Algorithmus eingesetzt.

Wird der Modulus q des Schemas als Zweierpotenz gewählt, ermöglicht der Aufbau des geheimen Schlüssels ein Verfahren, mit dessen Hilfe sich jede ganze Zahl $\in Z_q$ aus einem Geheimtext entschlüsseln lässt. In diesem Fall ist $q = 2^{l-1}$, die ersten $l-1$ Elemente des geheimen Schlüssels $\mathbf{v} = \mathsf{PowersOfTwo}_q(1, -\mathbf{s})$ sind $[1, 2, \ldots, 2^{l-2}]$.

Da für einen Geheimtext $\mathbf{C} = \mathsf{Enc}_{\mathcal{E}}(pk, \mu)$ gilt, dass $\mathbf{C} \cdot \mathbf{v} = \mu \cdot \mathbf{v} + noise$ mit $noise \ll q$, sind die ersten $l-1$ Koeffizienten von $\mathbf{C} \cdot \mathbf{v}$ entsprechend $\mu \cdot (1, 2, \ldots, 2^{l-1}) + noise$. Folgender Algorithmus rekonstruiert μ aus den ersten $l-1$ Elementen von $\mathbf{C} \cdot \mathbf{v}$ indem pro Vektorelement das least significant bit von $2^0 \cdot \mu, 2^1 \cdot \mu, \ldots$ bestimmt wird:

Algorithmus 1 : $\mathrm{MPDec}_q(\mathbf{v}, \mathbf{C})$

Eingabe : $\mathbf{v} \in \mathbb{Z}_q^N, \mathbf{C} \in \{0,1\}^{N \times N} = \mathsf{Enc}_{\mathcal{E}}(\mathbf{v}, \mu)$ mit $N = (n+1)\lfloor \log q \rfloor + 1$
Ausgabe : $\mu \in \mathbb{Z}_q$

```
1  result = 0;
2  b = C · v;
   // check first l-1 entries of b
3  for i ← (l − 2) to 0 do
4  |   x = b[i];
   |   // remove everything except current bit:
5  |   x = x − result ≪ j;
6  |   if x is small relative to q then
7  |   |   current bit = 0
8  |   end
9  |   else
10 |   |   current bit = 1, add to result
11 |   end
12 end
```

5.2.2 Homomorphe Auswertung eines *Branching Program*

Das in Kapitel 4.5.2 beschriebene asymmetrische Fehlerwachstum während der Multiplikation lässt sich nutzen, um das Fehlerwachstum einzuschränken und Systeme zu entwickeln, welche tiefere Boolesche Schaltkreise auswerten können. In der Implementierung wurde die Idee, Schaltkreise als Branching Program auszuwerten, aus [BV14] umgesetzt. Konkret kann ein System mit Dimensionsparameter n auf diese Art Schaltkreise der Tiefe $L = k \cdot \log n$ für eine beliebige Konstante k homomorph auswerten [BV14].

Für die homomorphe Auswertung wird ein Schaltkreis im ersten Schritt in ein *Permutation Branching Program der Breite 5* (siehe Abbildung 2.2) überführt. Ein Branching Program ist ein Berechnungsmodell zur Auswertung Boolescher Funktionen und wird folgendermassen aufgebaut:

Permutation Branching Program Ein Permutation Branching Program der Länge L mit einer Eingabe von $\{0,1\}^l$ ist eine Sequenz von L Tupeln $(var(t), \sigma_{t,0}, \sigma_{t,1})$, wobei $var(t)$ das Input-Bit bezeichnet, das im t-ten Tupel ausgewertet wird und $\sigma_{t,0}, \sigma_{t,1}$ zyklische Permutationen mit 5 Elementen beschreiben. Die Berechnung des Programms für einem Input $\mathbf{x} = (x_1, \ldots, x_l)$ funktioniert wie folgt: Der Berechnungszustand zu einem Zeitpunkt t ist eine Zahl $S_t \in \{1,2,3,4,5\}$, der Ausgangszustand ist $S_0 = 1$. Der Zustand S_t wird rekursiv berechnet als

$$S_t = \sigma_{t,var(t)}(S_{t-1})$$

Es wird also schrittweise das von $var(t)$ bezeichnete Input-Bit x_i betrachtet und auf dem vorherigen Zustand S_{t-1} für $x_i = 0$ die Permutation $\sigma_{t,0}$ ausgeführt beziehungsweise für $x_i = 1$ entsprechend die Permutation $\sigma_{t,1}$, um den neuen Zustand S_t zu erreichen. Nach L Berechnungsschritten ist das Programm im Endzustand S_L. Stimmt der mit dem anfangs festgelegten Zielzustand überein, gibt das Programm TRUE aus, sonst FALSE.

Um das Fehlerwachstum während der homomorphen Evaluation zu kontrollieren, wird der Zustand S als 5-dimensionaler Bitvektor $\mathbf{v}_t$ dargestellt, der die Zustände $1, 2, \ldots$ als $(10000), (01000), \ldots$ abbildet. Die Idee zur Auswertung eines Branching Programs auf verschlüsselten Daten ist, dass $\mathbf{v}_t[i] = 1$ nur dann gilt, wenn $\sigma_{t,var(t)}(S_{t-1}) = i$. Umgekehrt bedeutet das, dass $\mathbf{v}_t[i] = 1$ wenn

$$\begin{aligned} &\mathbf{v}_{t-1}[\sigma_{t,0}^{-1}] = 1 \text{ und } x_{var(t)} = 0 \qquad \text{oder} \\ &\mathbf{v}_{t-1}[\sigma_{t,1}^{-1}] = 1 \text{ und } x_{var(t)} = 1 \end{aligned}$$

Es gilt also für $t = 1, \ldots, L$ und $i \in \{1, 2, 3, 4, 5\}$ [BV14]:

$$\mathbf{v}_t[i] = \mathbf{v}_{t-1}[\sigma_{t,0}^{-1}(i)] \cdot (1 - x_{var(t)}) + \mathbf{v}_{t-1}[\sigma_{t,1}^{-1}(i)] \cdot x_{var(t)} \tag{5.1}$$

Die Werte für $\sigma_{t,0}^{-1}$ und $\sigma_{t,0}^{-1}$ lassen sich als Konstanten aus der Beschreibung des Branching Programs berechnen, die Gleichung 5.1 lässt sich also für $t = 1, \ldots, L$ und $i \in \{1, 2, 3, 4, 5\}$ homomorph mit einer verschlüsselten Version von $\mathbf{v}_t$ und verschlüsselten Input-Bits $x_1, \ldots, x_l$ auswerten.

Der Fehler in den Verschlüsselungen von $\mathbf{v}_t$, die den aktuellen Zustand während der Auswertung darstellen, wächst pro Auswertungsschritt um maximal $2 \cdot \sqrt{n \log q}$. Der Fehler nach der Auswertung eines Programms mit einer Tiefe von L und einem anfänglichen Rauschen von $noise(C_0)$ ist demnach kleiner als $2L \cdot \sqrt{n \log q} \cdot noise(C_0)$.

5.3 Parameterwahl

Die Wahl geeigneter Parameter q, n und χ für die Instanziierung des Systems stellte sich als sehr schwierig heraus. Um einschätzen zu können, in welcher ungefähren Größenordnung konkrete Parameter eines LWE Systems liegen, wurden Arbeiten zusammengestellt, welche die Sicherheit LWE basierter Systeme gegenüber konkreten Angriffen untersuchen, und basierend auf den Ergebnissen konkrete Parameterwerte vorschlagen [LP11, MR09, LNV11]. Einen Überblick bietet Tabelle 5.1.

Wie sich zeigte, ließen sich die ermittelten Beispielparameter nicht auf das GSW System übertragen, fast alle Settings erlaubten nicht eine fehlerfreie homomorphe Evaluation. Um die Auswertung möglichst tiefer Schaltkreise zu ermöglichen, muss q im Vergleich zu n und dem maximalen Fehler B deutlich größer gewählt werden, als in den Beispielen. Für der Implementierung von [BV14] wurde $q = n^4$ gewählt und der maximale Fehler passend für die gewünschte Tiefe des auszuwertenden Branching Programs bestimmt. Verletzen die resultierenden Parameter die Sicherheitsanforderungen des Systems, erzeugt das eine Exception.

In der Implementierung von [GSW13] ließen sich mit den im Paper vorgeschlagenen Parametersettings (siehe Tabelle 4.1) keine Systeme konstruieren, die Schaltkreise der gewünschten Tiefe korrekt auswerten konnten. Da hier jede homomorphe Auswertung den Fehler um einen Wert polynomiell in n vergrößerte, ließ sich mehr als eine homomorphe Auswertung nur ermöglichen, wenn die Sicherheitsanforderung, dass q polynomiell in n beschränkt sein muss, verletzt wurde.

Setup	n	q	stddev	B	N	L [GSW13]	L [BV14]
LP11 toy	128	2053	2,70	10,80	1548	0	0
LP11 low	192	4093	3,54	14,16	2316	0	0
LP11 medium*	256	4093	3,33	13,32	3084	0	0
LP11 high	320	4093	3,19	12,76	3852	0	0
MR09 low	136	2003	5,19	20,76	1507	0	0
MR09 high	214	16381	2.94	11,76	3010	0	1
LNV11*	2048	2^{58}	8	32	118842	2	$\approx 2^{40}$

Tabelle 5.1: Mögliche Systemparameter für ein LWE System mit Dimension n, modulus q, und einer diskreten Gaussverteilung mit Standardabbweichung *stddev* als Fehlerverteilung und einem maximalen Fehlerbetrag von B [LP11, MR09, LNV11]. Die mit $*$ gekennzeichneten Settings besitzen nach Aussage der Autoren ein mit AES-128 vergleichbares Sicherheitsniveau. Die rechte Seite der Tabelle betrachtet ein GSW System mit den gegebenen Parametern. N bezeichnet die Dimension der resultierenden Geheimtext-Matrizen, L gibt an, wie viele aufeinanderfolgende homomorphe Evaluationen das System korrekt ausführen kann.

5.4 Speicherbedarf

Ein gemeinsames Problem homomorpher Verschlüsselungsverfahren sind die großen Schlüssel und die Expansion der Geheimtexte. Für das GSW System gilt:

Geheimer Schlüssel: $sk \in \mathbb{Z}_q^{(n+1)\cdot\log q}$ mit einer Größe von $(n+1)\cdot\log^2 q$ Bits oder $O(n\log^2 q) = \tilde{O}(n)$.

Öffentlicher Schlüssel: $pk \in \mathbb{Z}_q^{m\times(n+1)}$ mit einer Größe von $2n\cdot\log^2 q\cdot(n+1) = (n^2+n)\cdot 2\log^2 q$ Bits oder $O(n^2\log^2 q) = \tilde{O}(n^2)$.

Verschlüsselung eines Bits: $\mathbf{C} \in \mathbb{Z}_q^{N\times N}$ mit einer Größe von $(n^2+2n+1)\cdot\log^3 q$ Bits oder $O(n^2\log^3 q) = \tilde{O}(n^2)$.

Der öffentliche Schlüssel und die Geheimtexte haben in einem LWE System normalerweise eine Größe von $\tilde{O}(n^3)$, diesen Wert kann das GSW System signifikant senken. Andererseits existieren auf anderen Technologien basierende Systeme, die deutlich effizienter arbeiten.

Tabelle 5.2 bietet einen Überblick über den tatsächlichen Speicherbedarf des implementierten Schemas mit unterschiedlichen Parametern. Sie zeigt die Größe des privaten Schlüssels, des öffentlichen Schlüssels und die Menge an Daten, die erzeugt wird, um ein Klartext-Bit zu verschlüsseln.

n	q	Private Key	Public Key	Ciphertext
16	16^4	0.56kB	12.75kB	163.12kB
32	32^4	1.69kB	72.18kB	1.14MB
64	64^4	4.76kB	414.37kB	7.55MB
128	128^4	12.78kB	2.20MB	46.71MB
256	256^4	33.12kB	11.54MB	274.38MB
512	512^4	83.41kB	56.69MB	1.51GB

Tabelle 5.2: Speicherbedarf eines GSW Systems mit Giterdimension n und Modulus q. Ciphertext bezeichnet den Geheimtext, der erzeugt wird, um ein Klartext-Bit zu verschlüsseln.

5.5 Laufzeitanalyse

Die Funktionen zur Ver- und Entschlüsselung entsprechen im Wesentlichen den von Regev entwickelten Algorithmen für ein LWE basierte Verschlüsselungssystem [Reg09], die Komplexität beträgt wie in anderen LWE basierten Systemen $\tilde{O}(n^3)$ für die Verschlüsselung und $\tilde{O}(n^2)$ für die Entschlüsselung eines Bits.

Die Auswertung eines NAND-Gatters wird dominiert von der Multiplikation zweier $N \times N$-Matrizen. Werden die Systemparameter so gewählt, wie in Tabelle 4.1 vorgeschlagen, gilt $N = O(n \cdot L \log n) = \tilde{O}(nL)$. Für die homomorphe Auswertung eines Gatters lässt sich dann das folgende Theorem aufstellen:

Theorem 5.1 ([GSW13]). *Für einen Dimensionsparameter n und einen Tiefenparameter L wertet das Schema NAND-Schaltkreise mit $\tilde{O}(nL)^\omega$ Operationen pro Gatter aus, wobei $\omega < 2.3727$ den Koeffizienten der Matrixmultiplikation darstellt.*

Im Vergleich zu früheren LWE basierten homomorphen Verschlüsselungsverfahren [Bra12, BV11a, BV11b], in denen die Komplexität einer homomorphe Berechnung pro Gatter mindestens $\tilde{O}(n^3 L)$ beträgt, ist der Wert asymptotisch besser.

5.5.1 Laufzeitmessung

In der Testumgebung, einem 2,4 GHz Intel Core i5 CPU mit 8 GB DDR3 RAM, benötigte das System für Schlüsselerzeugung, Ver- und Entschlüsselung und homomorphe Operationen die in Tabelle 5.3 gezeigten Zeiten.

Abbildung 5.1 und 5.1 stellen die gemessenen Laufzeiten grafisch dar.

Implementierung nach [GSW13]						
n	L	KeyGen	Enc	Dec	Add	Mult
16	1	0.03	3.00	0.00	1.53	1.51
20	1	0.06	6.05	0.01	3.05	3.06
24	1	0.12	11.43	0.00	5.71	5.66
28	1	0.19	19.36	0.00	9.83	9.74
32	1	0.10	32.19	0.01	16.10	16.22
64	1	0.62	350.70	0.04	133.14	133.91
128	1	4.86	-	-	-	-
Implementierung nach [BV14]						
n	L	KeyGen	Enc	Dec	Add	Mult
16	10	0.02	4.63	0.00	3.75	8.22
20	10	0.02	10.55	0.00	9.37	19.76
24	10	0.03	23.04	0.00	21.52	44.31
28	10	0.04	44.09	0.00	40.67	76.65
32	10	0.07	61.38	0.00	56.88	123.36
64	10	0.59	865.84	0.03	796.75	2878.05
128	10	4.01	-	-	-	-

Tabelle 5.3: Vergleich der Laufzeiten der Implementierungen nach [GSW13] und [BV14]. Verglichen wurden Systeme der Dimension n mit Modulus $q = n^4$. Benötigte Zeit für Schlüsselerzeugung, Ver- und Entschlüsselung sowie homomorphe Addition und Multiplikation in s.

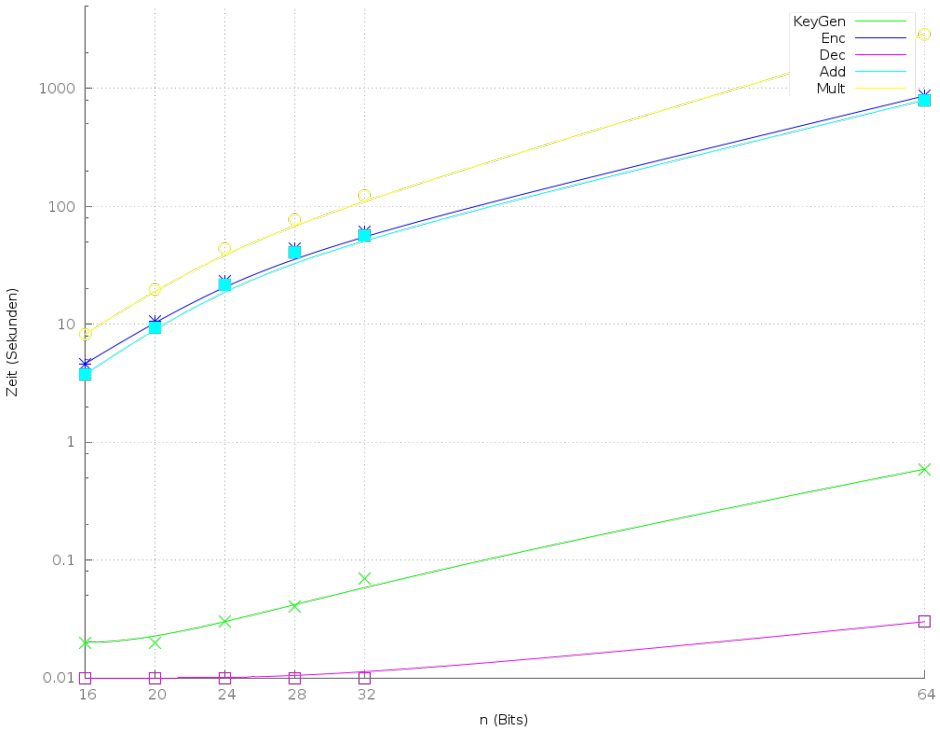

Abbildung 5.1: Laufzeitmessungen für [GSW13]

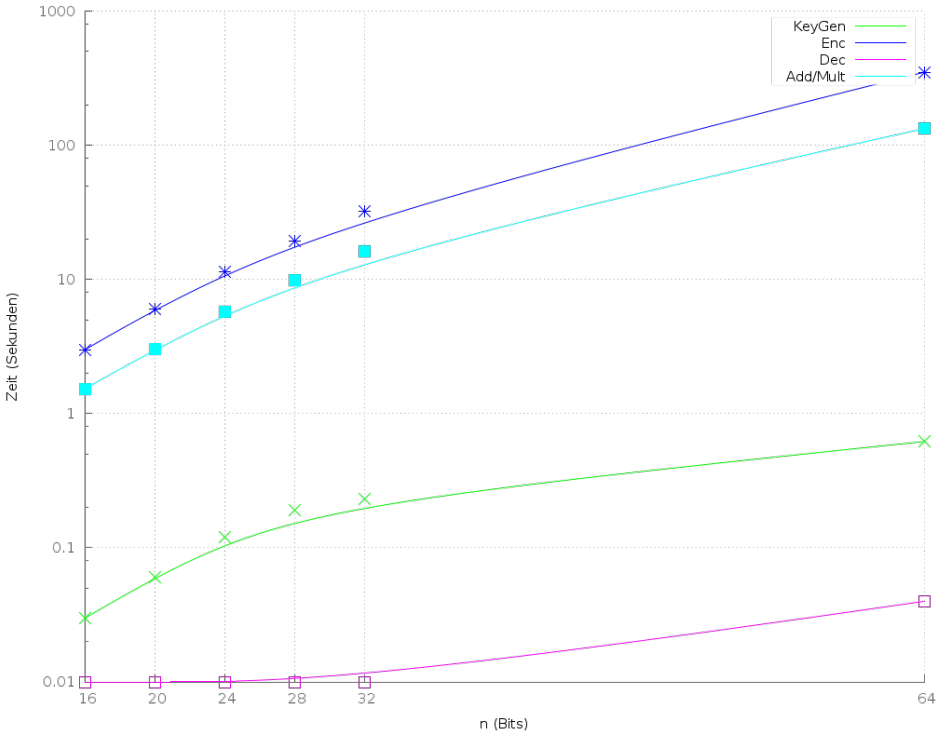

Abbildung 5.2: Laufzeitmessungen für [BV14]

Kapitel 6

Fazit

6.1 Zusammenfassung

Im Rahmen dieser Bachelor-Thesis wurde das Thema homomorphe Verschlüsselung unter verschiedenen Gesichtspunkten untersucht. Im ersten Teil der Arbeit wurde ein Überblick über den aktuellen State of the Art zusammengestellt, Motivationen für den Einsatz homomorpher Verschlüsselung und aktuelle Probleme aufgezeigt. Weiterhin wurden die zugrunde liegenden Ideen und Konzepte vorgestellt.

Den zweiten Teil der Arbeit bildete die Analyse und prototypische Implementierung eines aktuellen homomorphen Verschlüsselungssystems. Gewählt wurde das Approximate Eigenvector Schema von Gentry, Sahai und Waters von 2013 [GSW13], weil die geringere theoretische Komplexität der homomorphen Operationen verbesserte Laufzeiten gegenüber vergleichbaren Systemen versprach. Beim Approximate Eigenvector Schema handelt es sich um ein leveled homomorphes Verschlüsselungssystem, welches die Auswertung von Schaltkreisen der gewünschten Tiefe L nicht durch Bootstrapping, sondern durch die Wahl geeigneter Systemparameter ermöglicht. Dieser Ansatz stellte sich in der praktischen Umsetzung als problematisch heraus, da zur Auswertung tieferer Schaltkreise riesige Parameter nötig waren und die erforderlichen Parameterkombinationen die Sicherheit des Systems reduzierten.

Um die Auswertung tieferer Schaltkreise mit praktikableren Systemparametern zu ermöglichen, wurde das System ergänzt durch verschiedene Optimierungen von Brakerski und Vaikuntanathan [BV14]. Durch die Nutzung eines sequentiellen Auswertungsmodells für die zu berechnenden Booleschen Funktionen ließ sich das Fehlerwachstum während der homomorphen Auswertung deutlich reduzieren. Während der Fehler im ursprünglichen GSW System exponentiell mit der Tiefe L des ausgewerteten Schaltkreises anstieg, wächst der Fehler auf diese Art nur noch linear mit L und es ließ sich ein System realisieren, welches Schaltkreise der Tiefe $O(\log n)$

auswerten kann, wobei n den Dimensionsparameter des Systems bezeichnet.

Eine klare Schwäche des implementierten Systems ist die Geheimtextexpansion. Klartext-Daten wachsen während der Verschlüsselung um den Faktor $O(n^2 \cdot \log^3 n)$. Ring-LWE basierte System erzielen hier deutlich bessere Werte.

6.2 Ausblick

Aktuelle Implementierungen voll homomorpher Verschlüsselungssysteme arbeiten bereits deutlich effizienter als die ersten Systeme von 2010, trotzdem ist ihre Komplexität für den praktischen Einsatz noch deutlich zu hoch. Da der Markt für praktikable Lösungen als sehr groß eingeschätzt wird, ist das Interesse an homomorpher Verschlüsselung dessen ungeachtet groß, beispielsweise forschen IBM [IBM13], Microsoft [MR13] und Fujitsu [FLL13] an eigenen Verfahren.

Wichtige offene Fragen und Probleme auf dem Gebiet homomorpher Verschlüsselung sind Folgende [Bra13]:

Kurze Schlüssel ohne Circular Security Existierende FHE Systeme stellen unterschiedliche Varianten von Gentrys Blueprint dar. Folglich ist zur Konstruktion eines voll homomorphen Verschlüsselungssystems immer die Annahme der *Circular Security* des Systems nötig. Bisher existiert kein System, welches voll homomorphe Verschlüsselung ohne diese Annahme ermöglicht.

FHE mit neuen kryptographischen Annahmen Alle heute bekannten voll homomorphen Verschlüsselungssysteme stammen aus dem Gebiet der gitterbasierten Kryptographie, die Konstruktion eines FHE Systems, welches auf anderen Annahmen beruht, steht noch aus. Eine Hoffnung ist, dass durch ein grundsätzlich anderes Vorgehen praktikablere Systeme entstehen könnten.

Bounded Malleability *Non-Malleability* als Eigenschaft eines kryptographischen Systems verlangt, dass es nicht möglich ist, einen Geheimtext so zu verändern, dass daraus eine Verschlüsselung einer anderen Nachricht wird, also dass kein Angreifer den Inhalt einer verschlüsselten Nachricht verändern kann. Offensichtlich widersprechen sich die Forderungen nach Homomorphie und *Non-Malleability* eines Verschlüsselungssystems, da homomorphe Verschlüsselung gerade ermöglicht, einen Geheimtext $\mathbf{C}_1 = \mathsf{Enc}_{\mathcal{E}}(m)$ in einen anderen $\mathbf{C}_2 = \mathsf{Enc}_{\mathcal{E}}(f(m))$ zu überführen. Notwendig wäre ein System, das beide Eigenschaften kombiniert, so dass nur

Funktionen aus einer festgelegten Klasse homomorph berechnet werden können. Man nennt ein solches System *bounded malleable* [Vai11].

Verbesserte Effizienz Die vermutlich wichtigste Aufgabe in Bezug auf homomorphe Verschlüsselung ist die Suche nach effizienteren Systemen beziehungsweise die Optimierung bestehender Systeme.

6.3 Optimierungsmöglichkeiten

Als Standard Benchmark zur Leistungsmessung eines homomorphen Verschlüsselungssystems hat sich die homomorphe Auswertung der AES 128 Verschlüsselung durchgesetzt. Während aktuelle Implementierungen etwa 5 Minuten für die homomorphe Verschlüsselung eines Blocks benötigen [GHS12, CCK+13], waren es vor 4 Jahren noch 36 Stunden [GH10]. Neben der Optimierung der verwendeten Algorithmen gibt es aktuell folgende weitere Ansätze zur Konstruktion effizienterer Systeme:

Batching Eine Möglichkeit, sowohl die Expansion der Geheimtexte als auch die Komplexität der homomorphen Evaluation zu reduzieren besteht darin, statt eines einzelnen Bits mehrere Klartext-Nachrichten pro Geheimtext-Block zu codieren. Der Server kann auf diese Art eine homomorphe Operation parallel auf mehreren Eingaben ausführen [CCK+13].

Hybride voll homomorphe Verschlüsselung In voll homomrphen Systemen hat der Verschlüsselungsalgorithmus neben der homomorphen Auswertung die höchste Komplexität. Da diese auf dem Clientrechner stattfindet, stellt die aufwändige Verschlüsselung gemeinsam mit den großen Geheimtexten (siehe Tabelle 5.2) für einen Einsatz im Cloud Computing ein Problem dar.

Symmetrische, nicht homomorphe Verfahren wie AES ermöglichen eine effizientere Verschlüsselung, sowohl in Bezug auf die benötigte Zeit, als auch den Speicherplatz. Ein hybrider Ansatz versucht, die Vorteile beider Verfahren zu kombinieren.

Dazu überträgt der Nutzer eines Cloud Dienstes seine Daten symmetrisch verschlüsselt an den Cloud Server. Gleichzeitig stellt er diesem eine homomorph verschlüsselte Version $\mathsf{Enc}_{\mathcal{E}}(pk, sym)$ des symmetrischen Schlüssels sym zur Verfügung. Der Server kann jetzt die symmetrische Verschlüsselungsfunktion mit $\mathsf{Enc}_{\mathcal{E}}(pk, sym)$ und den symmetrisch verschlüsselten Daten als Eingabe homomorph auswerten

und erhält eine homomorph verschlüsselte Variante der eigentlichen Daten. Auf diesen lassen sich die gewünschten Berechnungen vornehmen [Bra13].

Literaturverzeichnis

[Bar86] D A Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc1. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, STOC '86, pages 1–5, New York, NY, USA, 1986. ACM.

[BCI+07] Julien Bringer, Hervé Chabanne, Malika Izabachène, David Pointcheval, Qiang Tang, and Sébastien Zimmer. An application of the goldwasser-micali cryptosystem to biometric authentication. In Josef Pieprzyk, Hossein Ghodosi, and Ed Dawson, editors, *Information Security and Privacy*, volume 4586 of *Lecture Notes in Computer Science*, pages 96–106. Springer Berlin Heidelberg, 2007.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In *Proceedings of the Second International Conference on Theory of Cryptography*, TCC'05, pages 325–341, Berlin, Heidelberg, 2005. Springer-Verlag.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 309–325, New York, NY, USA, 2012. ACM.

[BL14] Alexandra Berkoff and Feng-Hao Liu. Leakage resilient fully homomorphic encryption. In Yehuda Lindell, editor, *Theory of Cryptography*, volume 8349 of *Lecture Notes in Computer Science*, pages 515–539. Springer Berlin Heidelberg, 2014.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. *IACR Cryptology ePrint Archive*, 2012:78, 2012. informal publication.

[Bra13] Zvika Brakerski. Fully homomorphic encryption. http://www.ic.unicamp.br/ascrypto2013/slides/ascrypto2013_zvikabrakerski.pdf, 2013.

[BV11a] Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 97–106, Oct 2011.

[BV11b] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 505–524. Springer Berlin Heidelberg, 2011.

[BV14] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based fhe as secure as pke. In *Proceedings of the 5th Conference on Innovations in Theoretical Computer Science*, ITCS '14, pages 1–12, New York, NY, USA, 2014. ACM.

[CCK+13] JungHee Cheon, Jean-Sébastien Coron, Jinsu Kim, MoonSung Lee, Tancrède Lepoint, Mehdi Tibouchi, and Aaram Yun. Batch fully homomorphic encryption over the integers. In Thomas Johansson and PhongQ. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 315–335. Springer Berlin Heidelberg, 2013.

[CF85] Josh D. Cohen and Michael J. Fischer. A robust and verifiable cryptographically secure election scheme. In *Proceedings of the 26th Annual Symposium on Foundations of Computer Science*, SFCS '85, pages 372–382, Washington, DC, USA, 1985. IEEE Computer Society.

[FLL13] Fujitsu Laboratories Ltd. Fujitsu develops world's first homomorphic encryption technology that enables statistical calculations and biometric authentication. http://www.fujitsu.com/global/about/resources/news/press-releases/2013/0828-01.html, 2013.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 169–178, New York, NY, USA, 2009. ACM.

[Gen10] Craig Gentry. Computing arbitrary functions of encrypted data. *Commun. ACM*, 53(3):97–105, March 2010.

[GH10] Craig Gentry and Shai Halevi. Implementing gentry's fully-homomorphic encryption scheme. Cryptology ePrint Archive, Report 2010/520, 2010. http://eprint.iacr.org/.

[GHS12] Craig Gentry, Shai Halevi, and NigelP. Smart. Homomorphic evaluation of the aes circuit. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 850–867. Springer Berlin Heidelberg, 2012.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and JuanA. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92. Springer Berlin Heidelberg, 2013.

[IBM13] IBM. Made in ibm labs: Advancing privacy and security in the cloud. http://www-03.ibm.com/press/us/en/pressrelease/42808.wss, 2013.

[KL07] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series)*. Chapman & Hall/CRC, 2007.

[KM03] H.J. Kowalsky and G.O. Michler. *Kowalsky/michler:lineare Algebra 12a Lg*. De-Gruyter-Lehrbuch. de Gruyter, 2003.

[KO97] Eyal Kushilevitz and Rafail Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval (extended abstract). In *IN PROC. OF THE 38TH ANNU. IEEE SYMP. ON FOUNDATIONS OF COMPUTER SCIENCE*, pages 364–373, 1997.

[LATV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, STOC '12, pages 1219–1234, New York, NY, USA, 2012. ACM.

[LNV11] Kristin Lauter, Michael Naehrig, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? Cryptology ePrint Archive, Report 2011/405, 2011. http://eprint.iacr.org/.

[LP11] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for lwe-based encryption. In *Proceedings of the 11th International Conference on Topics in Cryptology: CT-RSA 2011*, CT-RSA'11, pages 319–339, Berlin, Heidelberg, 2011. Springer-Verlag.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718. Springer Berlin Heidelberg, 2012.

[MR09] Daniele Micciancio and Oded Regev. Lattice-based cryptography. In DanielJ. Bernstein, Johannes Buchmann, and Erik Dahmen, editors, *Post-Quantum Cryptography*, pages 147–191. Springer Berlin Heidelberg, 2009.

[MR13] Microsoft Research. Cloud security and cryptography. `http://research.microsoft.com/en-us/projects/cryptocloud/`, 2013.

[PBD05] Kun Peng, Colin Boyd, and Ed Dawson. A multiplicative homomorphic sealed-bid auction based on goldwasser-micali encryption. In Jianying Zhou, Javier Lopez, RobertH. Deng, and Feng Bao, editors, *Information Security*, volume 3650 of *Lecture Notes in Computer Science*, pages 374–388. Springer Berlin Heidelberg, 2005.

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: Extended abstract. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 333–342, New York, NY, USA, 2009. ACM.

[RAD78] R L Rivest, L Adleman, and M L Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation, Academia Press*, pages 169–179, 1978.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 84–93, New York, NY, USA, 2005. ACM.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):34:1–34:40, September 2009.

[Reg10] Oded Regev. The learning with errors problem (invited survey). In *IEEE Conference on Computational Complexity*, pages 191–204. IEEE Computer Society, 2010.

[Rot13] RonD. Rothblum. On the circular security of bit-encryption. In Amit Sahai, editor, *Theory of Cryptography*, volume 7785 of *Lecture Notes in Computer Science*, pages 579–598. Springer Berlin Heidelberg, 2013.

[S+09] W. A. Stein et al. Sage Mathematics Software (Version 6.1.1). http://www.sagemath.org, June 2009.

[SV09] N.P. Smart and F. Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. Cryptology ePrint Archive, Report 2009/571, 2009. http://eprint.iacr.org/.

[TT08] Gerald Teschl and Susanne Teschl. *Mathematik für Informatiker: Band 1: Diskrete Mathematik und Lineare Algebra*, volume 1 of *Mathematik für Informatiker*. Springer, zweite edition, Mai 2008.

[Vai11] V. Vaikuntanathan. Computing blindfolded: New developments in fully homomorphic encryption. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 5–16, Oct 2011.

[vDGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 24–43. Springer Berlin Heidelberg, 2010.

[Weg87] Ingo Wegener. *The Complexity of Boolean Functions*. John Wiley & Sons, Inc., New York, NY, USA, 1987.