**Hochschule Wismar**
University of Applied Sciences Technology, Business and Design
Fakultät für Ingenieurwissenschaften

# Master-Thesis

## Quantenresistente vertrauliche Cloud-Nutzung

Abschlussarbeit zur Erlangung des Grades eines

**Master of Engineering**

der Hochschule Wismar

| | |
|---|---|
| eingereicht von: | Linus Töbke |
| Erstgutachter: | Prof. Dr. Ing. habil. Andreas Ahrens |
| Zweitgutachter: | Prof. Dr. rer. nat. Nils Gruschka |

Hamburg, den 28. August 2022

## Danksagung

Mein erster Dank gilt Herrn Prof. Dr. Ing. habil. Andreas Ahrens für die Betreuung dieser Arbeit und der regelmäßigen Austausche während der Bearbeitung, die mir sehr geholfen haben. Mein zweiter Dank gilt Herrn Prof. Dr. rer. nat. Nils Gruschka für die Übernahme des Zweitgutachten und die Unterstützung bei den Überlegungen des Themas.

Zusätzlich möchte ich mich noch bei meinem aktuellen Arbeitgeber, der PPI AG, bedanken, da ich während des gesamten Studiums insbesondere für die Bearbeitung der Thesis ohne Probleme Urlaub nehmen konnte.

Auch meiner Freundin, Laura Hillemann, möchte ich für die seelische Unterstützung und das Korrekturlesen der Arbeit vom Herzen danken <3.

Ein indirekter Dank geht an meine Eltern, die durch die damalige Unterstützung in der Kindheit und Jugend den Weg für dieses Studium geebnet haben.

**Linus Töbke**

**Hinweise zur Lesbarkeit** Der Text ist hierarchisch in einem 4-Ebenen-Modell (z.B. 2.1.1.3) gegliedert. Auch, wenn dies für wissenschaftliche Arbeiten untypisch ist, dient es dem Leser als Hilfestellung, sodass bewusst bestimmte Unterabschnitte eine Ebene tiefer aufgeteilt wurden.

Aus Gründen der besseren Lesbarkeit wird auf eine geschlechtsspezifische Differenzierung, wie beispielsweise Angreifer:innen, verzichtet. Nichtsdestotrotz gelten entsprechende Begriffe stellvertretend für aller Geschlechter und im Sinne der Gleichbehandlung aller Geschlechter.

## Aufgabenstellung

Die Cloud ist seit mehreren Jahren in aller Munde und aus dem Alltag nicht mehr wegzudenken. Weder im privaten noch im beruflichen Kontext möchte man auf die Vorteile der Cloud-Nutzung verzichten. Allerdings bringt die Cloud auch Nachteile mit sich, da die Vertraulichkeit der Daten nicht gewährleistet ist. Lädt ein Nutzer beispielsweise seine Urlaubsbilder in die Cloud, dann kann der Cloud-Anbieter diese betrachten. Somit bedarf es einer Möglichkeit der vertraulichen Cloud-Nutzung, um die Vertraulichkeit der Daten zu gewähren.

Um dies zu ermöglichen, gibt es verschiedene Ansätze, wie beispielsweise die homomorphe Verschlüsselung. Die homomorphe Verschlüsselung erlaubt es, Berechnungen auf Geheimtexten (verschlüsselte Daten) durchzuführen, ohne dass das Ergebnis der Geheimtext-Berechnung von dem Ergebnis der Klartext-Berechnung abweicht. Entsprechend können die Vorteile der Cloud genutzt werden, während die Vertraulichkeit gewahrt werden kann. Dahingehend gib es in der Literatur bereits verschiedene Arbeiten, die die verschiedenen Ansätze miteinander vergleichen.

In den letzten Jahren bahnt sich aufgrund der rasanten Forschung eine neue Bedrohung für die klassischen Public-Key-Kryptosysteme an: Der Quantencomputer. Dieser berechnet die mathematischen Probleme der asymmetrischen Verfahren deutlich effizienter als klassische Computer, sodass die aktuell noch gängigen asymmetrischen Kryptosysteme, wie RSA, zukünftig nicht mehr als sicher gelten. Dahingehend schafft die Post-Quantum-Kryptographie, welche quantenresistente Verfahren bereitstellt, Abhilfe.

Aufgabe dieser Masterarbeit ist es nun den Vergleich für Ansätze durchzuführen, die die Vertraulichkeit wahren und quantenresistent sind. Dazu werden zwei Verfahren, die gitterbasierte Verschlüsselung und die sicherere Mehrparteienberechnung, implementiert und verglichen. Der Fokus dieser Arbeit liegt auf dem Vergleich der Kosten, die beim Cloud-Betrieb der jeweiligen Verfahren anfallen.

## Kurzreferat

Die Cloud-Nutzung erfreut sich hoher Beliebtheit, kann allerdings die Vertraulichkeit der Daten nicht gewährleisten. Um diesen Nachteil auszugleichen, können Verfahren wie homomorphe Verschlüsselung oder sichere Mehrparteienberechnung verwendet werden. Beide Ansätze lassen sich mit klassischen oder quantenresistenten Kryptosystemen umsetzen. Während es für klassische Kryptosysteme bereits Vergleiche der Ansätze gibt, wurde ein solcher Vergleich noch nicht für quantenresistente Ansätze durchgeführt.

In dieser Masterarbeit werden zwei Möglichkeiten der quantenresistenten vertraulichen Cloud-Nutzung vorgestellt: Die gitterbasierte Verschlüsselung und die sichere Mehrparteienberechnung. Beide Verfahren werden mit Hilfe der aktuellen Krypto-Bibliotheken in Form von zwei Prototypen verglichen. Der Fokus des Vergleichs liegt nicht - wie sonst in der Literatur üblich - lediglich auf der Effizienz der Durchlaufzeit, sondern auf dem Vergleich der Kosten für den Cloud-Betrieb der jeweiligen Verfahren. Die Arbeit zeigt verschiedene Szenarien und Kostenkalkulationen auf Basis verschiedener Cloud-Anbieter auf. Bei jedem dieser Vergleiche sticht die sichere Mehrparteienberechnung mindestens um einen Faktor von 28 die gitterbasierte Verschlüsselung hinsichtlich der Kosten.

## Abstract

Cloud-Computing enjoys high popularity but cannot guarantee data privacy. To compensate for this disadvantage, methods such as homomorphic encryption or secure multiparty computation can be used. Both approaches can be implemented with classical crypto or quantum-proofed crypto schemes. While there are already quite a few comparisons of the approaches for classical crypto schemes, such a comparison has not yet been done for quantum-proofed approaches.

In this master thesis, two options for quantum-proofed confidential cloud-computing, firstly lattice-based encryption, and secondly secure multiparty computation, were presented. Both methods were compared with each other using the latest crypto libraries in the form of two prototypes. The focus of the comparison was not, as is usually the case in the literature, merely on the efficiency of the runtime, but on the comparison of the costs for the cloud operation of the respective methods. This work shows different scenarios and cost calculations based on different cloud providers. In each of these comparisons, the secure multiparty computation outperforms the lattice-based encryption by computing at least 28th times cheaper.

# Inhaltsverzeichnis

# 1 Einleitung

Eine Studie des Vereins Bitkom hat ermittelt, dass 76% der mittelständischen Unternehmen in Deutschland Cloud-Computing nutzen. Die Unternehmen, die bislang noch nicht die Cloud nutzen, planen zukünftig den Einsatz der Technologie, ausgenommen von 6% der Unternehmen, die auch in Zukunft die Cloud nicht nutzen möchten [1]. Cloud-Computing hat dementsprechend bereits heute eine hohe Relevanz, die künftig weiter zunehmen wird. Nachteilig bei der Nutzung von Cloud-Computing ist, dass die Vertraulichkeit der Daten in der Cloud nicht gewährleistet ist. Der Cloud-Nutzer muss dem Cloud-Anbieter vertrauen, dass dieser beispielsweise die Daten nicht an ein Konkurrenz-Unternehmen verschickt. Wenn der Cloud-Nutzer die Vertraulichkeit der Daten sicherstellen möchte, dann muss der Cloud-Nutzer die Daten zuvor verschlüsseln, welches neue Herausforderungen mit sich bringt.

## 1.1 Problemstellung

Für die vertrauliche Cloud-Nutzung bedarf es der Verschlüsselung der Daten. Allerdings schränkt die Verschlüsselung die Nutzung der Cloud ein. Auf den verschlüsselten Daten können keine Berechnungen durchgeführt werden. Da viele Unternehmen jedoch die Daten in die Cloud auslagern, um Berechnungen durchzuführen, ist die vertrauliche Cloud-Nutzung eigentlich ausgeschlossen. Allerdings kann dieses Dilemma durch verschiedene mathematische Konstrukte aufgelöst werden. Abhilfe schafft beispielsweise die homomorphe Verschlüsselung mit Hilfe derer es möglich ist, auf Geheimtexten Rechenoperationen durchzuführen. So wurden in der Vergangenheit Protokolle entworfen, die aus einer Kombination von verschiedenen asymmetrischen Kryptoverfahren eine voll-homomorphe Verschlüsselung entwickelt haben. Diese ermöglichen eine vertrauliche Cloud-Nutzung. Ein Beispiel dafür ist das Multiprotokoll-Verschlüsselungs-Gateway von Bouti [2].

Durch die Entwicklung der Quantencomputer ist die Sicherheit der verwendeten asymmetrischen Kryptoverfahren gefährdet, da bereits in den 90ern Algorithmen

vorgestellt wurden, die die mathematischen Probleme der asymmetrischen Kryptoverfahren in annehmbarer Zeit lösen können [3]. Auch wenn die Forschung aktuell noch kein asymmetrisches Verfahren mit entsprechender Schlüssellänge brechen konnte, ist davon auszugehen, dass zukünftig die Verfahren gebrochen werden. Daher stellt die Verwendung der aktuellen asymmetrischen Verfahren schon jetzt ein potenzielles Risiko dar, da ein Angreifer die Daten heute speichern und in der Zukunft entschlüsseln kann [4]. Daher ergibt sich eine weitere Anforderung an die Nutzung der Cloud: Sie muss nicht nur vertraulich, sondern auch den Quantencomputern Stand halten, bzw. quantenresistent sein.

## 1.2 Ziel

Um die quantenresistente und vertrauliche Nutzung der Cloud zu ermöglichen, sollen zwei verschiedene Verfahren in der Thesis untersucht werden: Die homomorphe Verschlüsselung auf Basis der gitterbasierten Verschlüsselung und die sichere Mehrparteienberechnung auf Basis von Garbled Circuits. Beide Verfahren sind quantenresistent und erlauben die vertrauliche Nutzung der Cloud. Bislang wurde in der Literatur lediglich ein Vergleich beider Verfahren hergestellt, ohne die Quantenresistenz zu berücksichtigen. Zudem fokussieren sich die meisten wissenschaftlichen Ausarbeitungen lediglich auf die Durchlaufzeit der Verfahren. Diese Thesis hat den Anspruch vor allem die anfallenden Kosten im Cloud-Betrieb zu beleuchten, da diese in wirtschaftlichen Abwägungen immer eine große Rolle spielen. Zusammenfassend ist das Ziel der Thesis die Beantwortung der folgende Forschungsfrage:

> „Welches Verfahren ist im Cloud-Betrieb preiswerter: Die gitterbasierte homomorphe Verschlüsselung oder die sichere Mehrparteienberechnung?“

## 1.3 Methode

Die beiden Verfahren unterscheiden sich insofern, dass die homomorphe Verschlüsselung rechenbasiert und die Mehrparteienberechnung netzwerkbasiert ist. Um die Forschungsfrage beantworten zu können, wird für beide Methoden jeweils ein Prototyp implementiert. Es wird mit Hilfe der wissenschaftlichen Methode „Experiment“ die Forschungsfrage untersucht. Diese Methode wurde gewählt, da die Forschungsfrage einen starken Praxisbezug hat, welcher sich am besten in einem Experiment untersuchen lässt.

## 1.4 Aufbau

Die Thesis beginnt mit dem Erläutern der notwendigen Grundlagen in Kapitel 2. In dem Kapitel werden zunächst die kryptographischen Grundlagen geschaffen, indem die Post-Quantum-Kryptographie und die homomorphe Verschlüsselung vorgestellt werden. Da die Forschungsfrage auf den Vergleich zweier Verfahren abzielt, werden in den nächsten Abschnitten beide Verfahren (gitterbasierte Verschlüsselung und sichere Mehrparteienberechnung) beschrieben. Dabei wird der Fokus auf die jeweiligen Problemstellungen und den Bezug zur Quantenresistenz, sowie Vertraulichkeit gelegt. Da sich die Forschungsfrage auf Cloud-Computing bezieht, werden im folgenden Abschnitt die verschiedenen Eigenschaften, Modelle und Architekturen erläutert. Das Grundlagen-Kapitel schließt mit einer Zusammenfassung der zuvor geschilderten Grundlagen ab, indem die beiden Verfahren zur quantenresistenten vertraulichen Cloud-Nutzung verglichen werden. Somit stellt das Kapitel 2 alles für den Leser bereit, um die theoretischen Überlegungen für die Forschungsfrage nachvollziehen zu können.

Das Kapitel 3 geht auf die Architektur und Implementierung der jeweiligen Prototypen ein. Dabei wird zunächst der homomorphe Prototyp und anschließend der Prototyp der Mehrparteienberechnung vorgestellt. Danach werden die beiden Prototypen kritisch beleuchtet, um Verbesserungs- bzw. Ausbaumöglichkeiten und Grenzen aufzuzeigen. Das Kapitel schließt mit den Bewertungskriterien für die Evaluation der Verfahren ab. Zusammenfassend behandelt das dritte Kapitel alle praktischen Aspekte beider Prototypen, um für den Leser erkenntlich zu machen, auf welcher Grundlage die Evaluation und damit die Beantwortung der Forschungsfrage aufgebaut wird.

Das Kapitel 4 beantwortet die Forschungsfrage, da in diesem der eigentliche Vergleich durchgeführt wird. Dazu werden zunächst die Rahmenbedingungen für den Vergleich geschildert. Anschließend werden verschiedene Szenarien vorgestellt und die Ergebnisse der Durchführung aufgeführt. Das Kapitel schließt mit einer Bewertung für alle Szenarien ab und liefert eine „Bierdeckelrechnung“, die für eine schnelle Indikation des preiswerteren Verfahrens sorgt. Welches Verfahren preiswerter ist, hängt von den Kostenparametern des Cloud-Anbieters ab.

Die Thesis schließt mit dem Fazit in Kapitel 5 ab. Das Fazit fasst die gesamte Arbeit zusammen und geht nochmals auf die Beantwortung der Forschungsfrage ein.

# 2 Grundlagen

In diesem Kapitel werden die notwendigen Grundlagen in Form von Definitionen und Zusammenhängen geschaffen. Zu Beginn werden die verschiedenen kryptographischen Verfahren vorgestellt. Anschließend werden die gitterbasierten Verfahren und die Mehrparteienberechnungen detaillierter erläutert. Da die ausgesuchten Verfahren hinsichtlich der Cloud untersucht werden, bedarf es einer Erörterung der Grundlagen des Cloud-Computings. Abschließend werden die beiden Verfahren und Cloud-Computing zusammengebracht, sodass die notwendigen theoretischen Grundlagen für die Beantwortung der Forschungsfrage geschaffen sind. Es werden grundlegende Kenntnisse der Kryptographie und linearen Algebra vorausgesetzt.

**Notation** In der Thesis wird die folgende Notation verwendet: Vektoren werden als kleine Buchstaben und mit einem überliegenden Pfeil nach rechts gekennzeichnet (Beispiel: $\vec{s} \in \mathbb{Z}_q^n$ ). Matrizen werden als Großbuchstaben, die fett gedruckt werden, gekennzeichnet (Beispiel: $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$). Gitter werden als Großbuchstaben, die kursiv gedruckt werden, gekennzeichnet (Beispiel $G(\vec{b}_1, ..., \vec{b}_\mathrm{m})$). Gleitkommazahlen werden mit Rundungsklammern gekennzeichnet, die entweder aufrunden ($\lceil z \rceil$), abrunden($\lfloor z \rfloor$) oder zur nächsten Ganzzahl runden ($\lfloor z \rceil$). Die Modulo-Operation wird entweder über $m \mod p$ oder durch das Tiefstellen (Beispiel:$[\mathrm{c}]_2$) gekennzeichnet.

## 2.1 Kryptographische Grundlagen

In diesem Abschnitt werden die notwendigen kryptographischen Grundlagen geschaffen. Dazu wird zunächst die Post-Quantum-Kryptographie vorgestellt. Anschließend wird das Konzept der homomorphen Verschlüsselung inklusive der typischen Unterformen (teil-,somewhat- und voll-homomorph) erläutert, welches unabdingbar für die im Prototypen ausgewählte gitterbasierte Verschlüsselung ist. Die Kenntnis über die grundlegenden Funktionsweisen der symmetrischen und asymmetrischen Verschlüsselung wird vorausgesetzt.

### 2.1.1 Post-Quantum-Kryptographie

Die Post-Quantum-Kryptographie beschäftigt sich mit kryptographischen Verfahren, die bei Angriffen von sowohl Quanten- als auch klassischen Computern die Vertraulichkeit wahren können. Im Unterschied zu der Quantenkryptographie können diese Verfahren auf klassischen Computern implementiert werden. Daher ist die Post-Quantum-Kryptographie aktuell von hoher Bedeutung. Denn ein Angreifer könnte die verschlüsselten Daten jetzt speichern und in der Zukunft, wenn die Entwicklung der Quantencomputer vorangeschritten ist, entschlüsseln [4]. Somit bietet die Post-Quantum-Kryptographie grundlegende Lösungsmodelle für die Beantwortung der Forschungsfrage. Grundsätzlich wird zwischen fünf verschiedenen Kategorien der Post-Quantum-Kryptographie unterschieden, die jeweils auf unterschiedlichen mathematischen Problemen beruhen: Multivariante, isogeniebasierte, hashbasierte, codebasierte und gitterbasierte Kryptographie. Um die Anforderungen aus der Problemstellung zu erfüllen, fokussiert sich diese Masterarbeit auf die gitterbasierte Kryptographie, da diese homomorphe Eigenschaften aufweist. Wie die homomorphe Verschlüsselung funktioniert, wird in dem nachfolgenden Abschnitt erläutert.

### 2.1.2 Homomorphe Verschlüsselung

Die homomorphe Verschlüsselung erlaubt das Rechnen auf Geheimtexten, welches den Berechnungen auf den entsprechenden Klartexten entspricht, indem der Homomorphismus auf ein kryptographisches System angewendet wird. Der Homomorphismus beschreibt die Strukturerhaltung einer Abbildung zweier gleichartiger algebraischen Strukturen. Dies führt zu der folgenden Definition [5]

Seien (X, $\perp_1$, ... , $\perp_n$) und (Y, $\top_1$, ..., $\top_n$) Mengen mit jeweils $n$ inneren Verknüpfungen. Eine Abbildung mit $f : X \to Y$ heißt Homomorphismus dieser Strukturen wenn für jedes Element von $a, b \in X$ und $1 \leq i \leq n$ folgendes gilt:

$$f(a \perp_i b) = f(a) \top_i f(b)$$

#### 2.1.2.1 Teil-homomorphe Verschlüsselungsverfahren

Teil-homomorphe Verschlüsselungsverfahren sind hinsichtlich einer Operation homomorph. So ist beispielsweise das Pallier-Kryptosystem additiv-homomorph [6], während RSA multiplikativ-homomorph ist [7]. Grundsätzlich werden die folgenden Eigenschaften bei teil-homomorphen Algorithmen ausgenutzt [2]:

**Formbarkeit** Ein Algorithmus ist formbar, wenn es möglich ist ohne Kenntnis über den Schlüssels $sk$ und Klartext $m$ einen gegebenen Geheimtext $c = Enc_{\text{pk}}(m)$ zu einem angepassten Geheimtext $c‘$ so umzuformen, dass mit Hilfe der Änderungsfunktion $f$ der Geheimtext $c‘$ entschlüsselt werden kann, da $f(m) = Dec_{\text{sk}}(c‘)$ gilt [8]. Daher wird die Formbarkeit häufig als Schwäche angesehen, da einem Angreifer so ein Chosen-Ciphertext-Angriff ermöglicht wird [9].

**Arithmetische Operation** Die homomorphe Eigenschaft wird zur Durchführung von arithmetischen Operationen auf Geheimtexten benutzt. Dabei werden verschiedene Geheimtexte kombiniert, um eine arithmetische Operation auf dem Klartext abzubilden.

Teil-homomorphe Verschlüsselungsverfahren ermöglichen das Ausführen von Operationen auf den Geheimtexten nur einem begrenzten Ausmaß, sodass nur bestimmte Schaltungen abgebildet werden können. Voll-homomorphe Verschlüsselungsverfahren hingegen können alle möglichen Schaltungen abbilden [10].

#### 2.1.2.2 Voll-homomorphe Verschlüsselungsverfahren

Aus der Definition des Homomorphismus und der Kenntnis über eine klassisches Verschlüsselungssystem kann die Definition für ein voll-homomorphes Verschlüsselungssystem abgeleitet werden [11]:

Ein homomorphes Verschlüsselungssystem $HE = (Keygen, Enc, Dec, Eval)$ ist ein 4-Tupel der folgenden Polynomialzeit-Algorithmen:

**Schlüsselerzeugung** Der Schlüsselerzeugungsalgorithmus $(pk, evk, sk) \leftarrow Keygen(1^{\text{K}})$ benötigt eine unäre Darstellung des Sicherheitsparameters $K$ als Eingabeparameter und generiert einen öffentlichen Schlüssel $pk$ zur Verschlüsselung, einen öffentlichen Auswertungsschlüssel $evk$ und einen privaten Schlüssel $sk$ zur Entschlüsselung.

**Verschlüsselung** Der Verschlüsselungsalgorithmus $c \leftarrow Enc_{\text{pk}}(m)$ benötigt den öffentlichen Schlüssel $pk$ und eine Ein-Bit-Nachricht $m \in \{0, 1\}$ als Eingabeparameter und gibt einen Geheimtext $c$ aus.

**Entschlüsselung** Der Entschlüsselungsalgorithmus $m \leftarrow Dec_{\mathrm{sk}}(c)$ benötigt den privaten Schlüssel $sk$ und einen Geheimtext $c$ als Eingabeparameter und gibt eine Nachricht $m \in \{0,1\}$ aus.

**homomorphe Auswertung** Der Algorithmus $c' \leftarrow Eval_{\mathrm{evk}}(C, \{c_1, ..., c_\ell\})$ benötigt den Auswertungsschlüssel $evk$, die Funktion $C$ mit $C : \{0,1\}^\ell \rightarrow \{0,1\}$ und eine Menge von $\ell$ Geheimtexten $\{c, ..., c_\ell\}$ und gibt den Geheimtext $c'$ aus.

Es lässt sich, ohne Informationen über die Eingangsparameter $\{m_1, ..., m_\ell\}$ und die Funktion $C$ zu verraten, der Geheimtext Geheimtext $c'$ (Ausgabe des Algorithmus $Eval$) zum Ergebnis der Funktion $C$ entschlüsseln. Es gilt [12]:

$$C(m_1, ..., m_\ell) = Dec(sk, Eval(evk, C, Enc(pk, m_1), ..., Enc(pk, m_\ell))) \tag{2.1}$$

**Formel 2.1:** homomorphe Auswertung

Die Abbildung 1 veranschaulicht den Aufbau eines voll-homomorphe Verschlüsselungsverfahren. Ein konkretes voll-homomorphes Verschlüsselungsverfahren wird in Kapitel 2.2.4 vorgestellt.

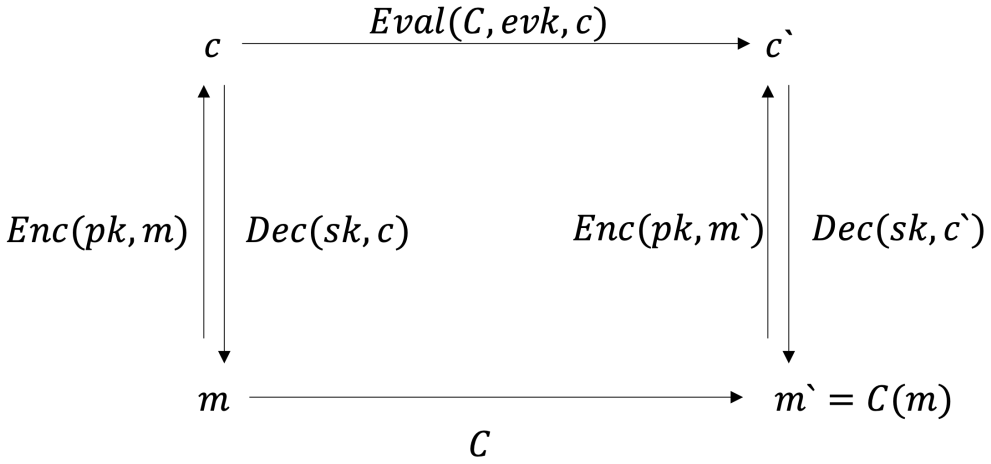



**Abbildung 1:** Generischer Aufbau eines voll-homomorphe Verschlüsselungsverfahren (Quelle: Eigene Zeichnung angelehnt an [5])

#### 2.1.2.3 Somewhat homomorphe Verschlüsselungsverfahren

Die somewhat homomorphe Verschlüsselung (SHE) ist eine Zwischenstufe zwischen der teil- und der voll-homomorphen Verschlüsselung. Bei der SHE können grundsätzlich im Gegensatz zur teil-homomorphen Verschlüsselung alle gegebenen Operationen homomorph berechnet werden. Der Unterschied zu der voll-homomorphen Verschlüsselung ist, dass nur eine kleine Anzahl an Berechnungen durchgeführt werden können, bevor das Rauschen zu groß wird und somit die Ergebnisse nicht mehr korrekt sind [13]. In Abschnitt 2.2.4.1 wird am Beispiel des Gentry Algorithmus auf Basis von Ganzzahlen genauer auf die Unterschiede eingegangen. Zudem lässt sich an dem Beispiel nachvollziehen, wie ein SHE Kryptosystem in ein FHE (Fully homomorphic encryption; deutsch: voll-homomorphes Kryptosystem) mit Hilfe des Bootstrapping transformiert werden kann.

In der Literatur findet man häufig noch das leveled homomorphe Verschlüsselungsverfahren (LHE). Dies beschreibt ein SHE, welches bis zu einem gewissen Limit (Tiefe von Schaltungen) korrekt funktioniert und kein Bootstrapping benötigt. Typischerweise ist dieses Limit deutlich höher bzw. tiefer als bei einem SHE [14]. Somit eignet sich ein LHE auch für praktische Anwendungen.

## 2.2 Gitterbasierte Verfahren

In diesem Abschnitt werden die notwendigen Grundlagen zu Gittern, Problemstellungen auf Gittern und gitterbasierte Verschlüsselungsverfahren geschaffen. Abschließend wird der Algorithmus von Gentry auf Basis der Ganzzahlen dargestellt, da dieser die Grundlage für das Kryptosystem darstellt, welches im Prototyp verwendet wurde.

### 2.2.1 Gitter

Ein Gitter wird durch n Vektoren im euklidischen Raum definiert. Formal gilt folgende Definition [15]:

Sei $\mathbb{R}^n$ der $n$-dimensionale euklidische Raum. Ein $n$-dimensionales Gitter $G$ in $\mathbb{R}^n$ ist das Set aus

$$G(\vec{b}_1, ..., \vec{b}_\mathrm{m}) = \{\sum_{i=1}^{N} x_\mathrm{i}\vec{b}_\mathrm{i} \mid x_\mathrm{i} \in \mathbb{Z}\} \tag{2.2}$$

**Formel 2.2:** Definition Gitter

linear unabhängigen Vektoren $\vec{b}_1, ..., \vec{b}_\mathrm{m}$ in $\mathbb{R}^\mathrm{n}(n \geqq m)$. Die Vektoren $\vec{b}_1, ..., \vec{b}_\mathrm{m}$ nennt man auch Basis des Gitters und werden in der Literatur oftmals als Spaltenvektoren in einer Matrix dargestellt:

$$\mathbf{B} = [\vec{b}_1, ..., \vec{b}_\mathrm{m}] \in \mathbb{R}^{\mathrm{n}\times\mathrm{m}}. \tag{2.3}$$

**Formel 2.3:** Definition Matrix

Setzt man $\mathbf{B}$ für $\vec{b}_\mathrm{i}$ mit $\{\ i \mid i \in 1, ..., m\ \}$ ein, dann erhält man eine kompaktere, äquivalente Definition:

$$G(\mathbf{B}) = \{\mathbf{B}x \mid x \in \mathbb{Z}^\mathrm{m}\}. \tag{2.4}$$

**Formel 2.4:** Definition Gitter (kompakt)

Die Abbildung 2 zeigt ein Gitter zur besseren Vorstellung eines Gitters. Das Gitter ist 2-dimensional und hat als Basis die Vektoren:

$$\vec{b}_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad \text{und} \quad \vec{b}_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Um besser zu verstehen, wie die Basis das Gitter definiert, ist in Abbildung 3 ein weiteres 2-dimensionales Gitter abgebildet, wobei hier der Vektor $\vec{b}_2$ das Gitter in x-Richtung „verschiebt“.

### 2.2.2 Problemstellungen auf Gittern

In diesem Abschnitt werden die grundlegenden Probleme dargestellt, die auf Gittern definiert sind und die Basis für gitterbasierte Kryptographie bilden.

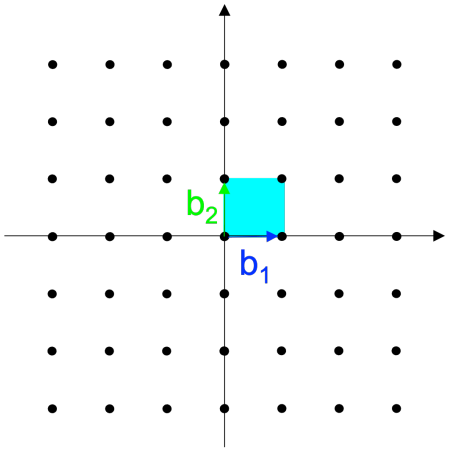



**Abbildung 2:** Abbildung eines Gitters $G \in \mathbb{Z}^2$ (Quelle: Eigene Zeichnung angelehnt an [16])

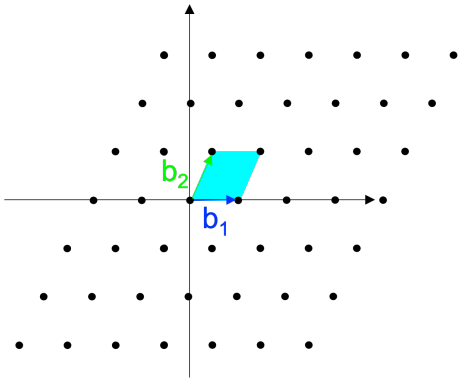



**Abbildung 3:** Abbildung eines Gitters $G \in \mathbb{Z}^2$ (Quelle: Eigene Zeichnung angelehnt an [16])

#### 2.2.2.1 Shortest Vektor Problem (SVP)

Das Shortest Vektor Problem (SVP) beschreibt ein Problem, welches durch das Finden des kürzesten Vektors (oder einen der kürzesten Vektoren) im Raum $\mathbb{R}^n$ gelöst wird [10]. Das Problem gibt es in drei Varianten, welche äquivalente Laufzeiten (Polynomialzeit) haben [17]:

Sei $G = G(\mathbf{B})$ ein beliebiges Gitter mit der Basis $\{ \mathbf{B}x \mid x \in \mathbb{Q}^{m \times n} \}$.

**Entscheidung** Gegeben sei zu der Basis $\mathbf{B}$ ein $d > 0$. Entscheide, ob:

$$\lambda_1(G(\mathbf{B})) \leq d \quad \text{oder} \quad \lambda_1(G(\mathbf{B})) > d \tag{2.5}$$

**Formel 2.5:** SVP (Variante Entscheidung)

**Berechnung** Berechne:

$$\lambda_1(G(\mathbf{B})) \tag{2.6}$$

**Formel 2.6:** SVP (Variante Berechnung)

**Suche** Finde einen nicht-trivialen Vektor $\vec{g} \neq 0 \in G(\mathbf{B})$, sodass gilt:

$$\| \vec{g} \| = \lambda_1(G(\mathbf{B})) \tag{2.7}$$

**Formel 2.7:** SVP (Variante Suche)

Die Abbildung 4 zeigt das SVP im 2-dimensionalen Raum. Dort scheint das Problem trivial. Die gesuchten bzw. kürzesten Vektoren werden mit dem ersten Blick auf das Gitter gefunden.

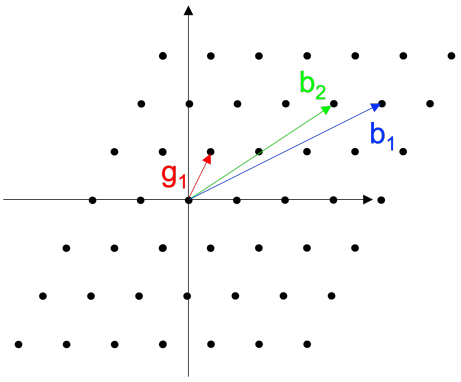



**Abbildung 4:** Abbildung des SVP; $\vec{b}_1$ und $\vec{b}_2$ als Basis und $\vec{g}_1$ als möglicher Lösung (Quelle: Eigene Zeichnung angelehnt an [18])

Für kryptographische Zwecke werden daher Gitter in viel höheren Dimensionen verwendet, um die Komplexität zu erhöhen. Des Weiteren werden zwischen guten und

schlechten Gitterbasen unterschieden. Gute Gitterbasen werden über Vektoren definiert, sind verhältnismäßig kurz und möglichst orthogonal zueinander sind. Schlechte Gitterbasen sind gegenteilig, also über Vektoren definiert, die lang und parallel zueinander sind. Die Abbildung 5 zeigt ein Gitter $G$, welches jeweils über zwei Basisvektoren verschiedener Basen definiert ist. Um ein Gefühl zu bekommen, warum die Basis $G(\mathbf{B})$ gut ist und die Basis $G(\mathbf{G})$ schlecht, hilft der Punkt $x_1$. Wenn der Punkt $x_1$ durch eine Linearkombination der Basisvektoren erreicht werden soll, ist es trivial zu erkennen, dass dies mit der Basis $G(\mathbf{B})$ deutlich einfacher ist. Diese Erkenntnis bezieht sich auf grundsätzlich auf gitterbasierte Verfahren und nicht nur das SVP.

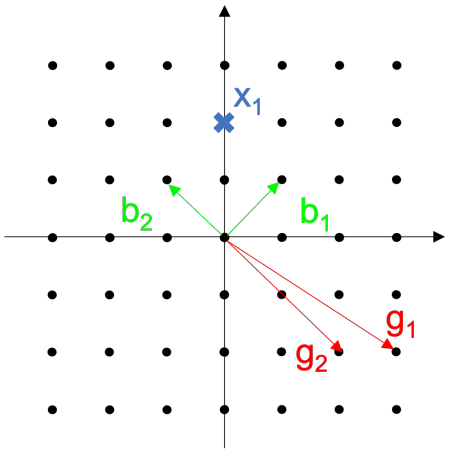



**Abbildung 5:** Abbildung zur Veranschaulichung von guten und schlechten Basen (Quelle: Eigene Zeichnung angelehnt an [17])

Die Komplexität des SVP gilt als NP-schwer [19]. In der Literatur wird zur Laufzeitbestimmung oftmals nicht das SVP, sondern das approximierte SVP verwendet. Das approximierte SVP beschreibt nicht die genaue Lösung des SVP, sondern eine Annäherung. Dazu wird die Definition um $\gamma$ (Näherungsfaktor) erweitert. Die Abbildung 6 zeigt das approximierte SVP aus Abbildung 4. Durch die Approximierung lassen sich Aussagen treffen, inwiefern die Komplexität von $\gamma$ abhängig ist. Die Tabelle 1 stellt die Komplexität und $\gamma$ ins Verhältnis. Es ist erkennbar, dass die Komplexitätsklassen von NP bis P reichen. Da das SVP bzw. Abwandelungen des SVP als Grundlage für verschiedene gitterbasierte Kryptographische Systeme genutzt werden, stellt die Schwere des Problems einen wichtigen Beitrag zu Sicherheit der Verfahren dar.

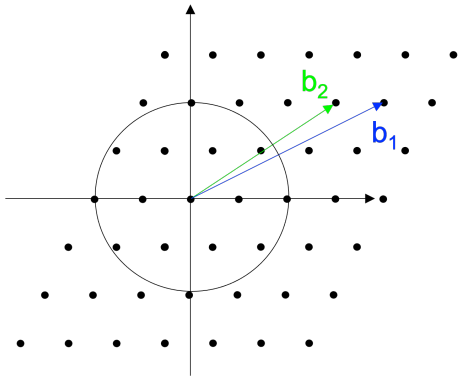



**Abbildung 6:** Abbildung des approximierten SVP; $b_1$ und $b_2$ als Basis und $\gamma = 2$ als Näherungsfaktor (Quelle: Eigene Zeichnung angelehnt an [18])

| **Näherungsfaktor $\gamma$** | **Komplexitätsklasse** | **gezeigt in** |
|---|---|---|
| 1 | NP | [19] |
| $2^{(\log n)^{1-e}}$ | NP | [20] |
| $O(\sqrt{n})$ | NP $\cap$ Co-NP | [21] |
| $2^{O(n)}$ | P | [22] |

**Tabelle 1:** Übersicht der Komplexitätsklassen im Bezug auf $\gamma$

#### 2.2.2.2 Closest Vektor Problem (CVP)

Das Closest Vektor Problem (CVP) beschreibt ein Problem, welches durch das Finden des nächsten (kürzesten Abstand) Gitterpunkts zu einem gegebenen Punkt im Raum $\mathbb{R}^n$ gelöst wird [15]. Wie beim SVP gibt es auch beim CVP drei Varianten, welche äquivalente Laufzeiten (Polynomialzeit) haben [17]:

Gegeben sei $G = G(\mathbf{B})$ ein beliebiges Gitter mit der Basis $\{ \mathbf{B}x \mid x \in \mathbb{Q}^{m \times n} \}$ und $\vec{v} \in \mathbb{Q}^n$.

**Entscheidung** Gegeben sei zu der Basis $B$ ein $r \in \mathbb{Q}$. Entscheide, ob:

$$dist(\vec{v}, (G(\mathbf{B}))) \leq r \quad \text{oder} \quad dist(\vec{v}, (G(\mathbf{B}))) > r. \tag{2.8}$$

**Formel 2.8:** SVP (Variante Entscheidung)

**Berechnung** Berechne:

$$dist(\vec{v}, (G(\mathbf{B}))) \tag{2.9}$$

**Formel 2.9:** SVP (Variante Berechnung)

**Suche** Finde einen Vektor $\vec{Bx}$ im Gitter $G$, sodass:

$$\| \vec{Bx} - \vec{v} \| \text{ minimal ist} \tag{2.10}$$

**Formel 2.10:** SVP (Variante Suche)

Die Abbildung 7 zeigt das CVP im 2-dimensionalen Raum. Genau wie beim SVP (siehe Abschnitt SVP) erscheint auch hier die Lösung trivial, sodass für kryptographische Zwecke deutlich höhere Dimensonen verwendet werden.

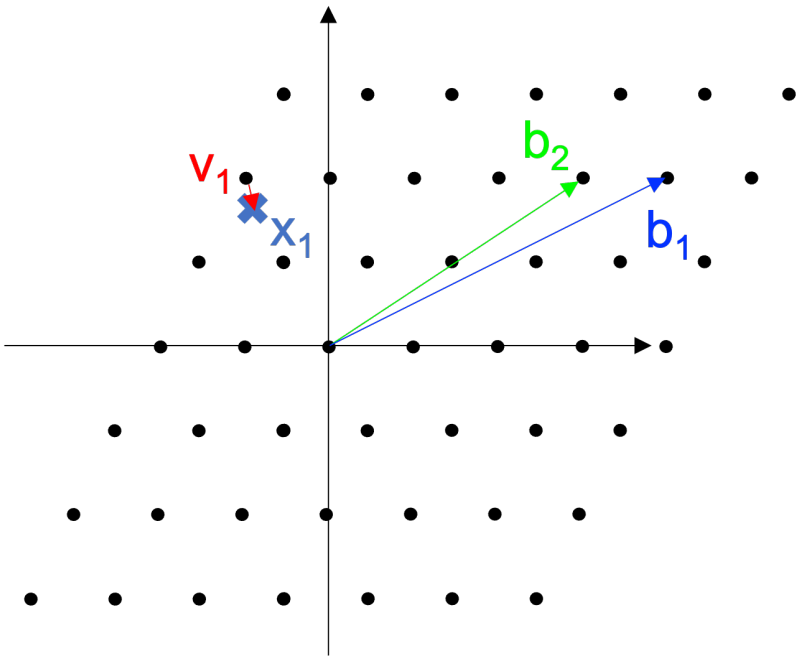



**Abbildung 7:** Abbildung des CVP; $\vec{b}_1$ und $\vec{b}_2$ als Basis und $\vec{v}_1$ als möglicher Lösung (Quelle: Eigene Zeichnung angelehnt an [18])

### 2.2.3 Gitterbasierte kryptographische Probleme

In diesem Unterabschnitt werden gitterbasierte Probleme vorgestellt, die als Grundlage für die meisten kryptographischen Systeme auf Gittern dienen. Dabei sind die vorgestellten Probleme die Brücke zwischen den allgemeinen Problemstellungen auf Gittern und gitterbasierten Public-Key Kryptosystemen.

#### 2.2.3.1 Small Integer Solution (SIS)

In seiner Pionierarbeit entdeckte Ajtai 1996, dass sich Gitter auch zur Konstruktion von Kryptosystemen eignen [23]. Dabei reduzierte Ajtai verschiedene approximierte Problemstellungen auf Gittern und entwickelte so den Vorläufer für das Problem, welches heutzutage als small integer solution Problem (SIS) bekannt ist. In dem SIS ist eine Menge von Vektoren $\vec{a}_1, \vec{a}_2, ...$ gegeben, die $\in \mathbb{Z}_q^n$ sind. Gesucht wird nach einer Teilmenge von Vektoren, die sich zu $0 \mod q$ aufaddiert [24]. Es wird also nach kurzen Vektoren in einem Gitter gesucht. Die Schwere des SIS-Problem wurde in [24] hinreichend untersucht: Die Lösung des SIS-Problems für jedes $q$, das mindestens ein Polynom in $n$ ist, impliziert eine Lösung für Standard-Gitterprobleme (z.B. SVP).

Das SIS stellt die Grundlage für kryptographische Primitive dar [25] und wird z.B. für kollisionsresistente Hash-Funktionen [26] und digitale Signaturen [27] benutzt.

#### 2.2.3.2 Learning with Errors (LWE)

Das Learning with Errors Problem (LWE) wurde in [28] vorgestellt und ist heutzutage die Grundlage für viele gitterbasierte Public-Key-Kryptosysteme. Beim LWE ist das Geheimnis $\vec{s} \in \mathbb{Z}_q^n$ gesucht, welches in einer Sequenz von „unscharfen“ linearen Gleichungen gegeben ist. Eine Sequenz könnte wie folgt aussehen (Beispiel übernommen von [25]):

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

Jede der Gleichungen ist bis auf eine kleine Unschärfe bzw. einen Fehler von $\pm 1$ korrekt. Ziel des Problems ist nun den Vektor $\vec{s}$ zu finden. In dem Beispiel ist die Lösung $\vec{s} = (0, 13, 9, 11)$. Ohne die Unschärfe wäre das Problem effizient über den Gauß-Algorithmus lösbar. Stellt man die „unscharfen“ linearen Gleichungen als Matrizen dar, dann ergibt sich folgendes Bild, wobei die erste Matrix ($\mathbb{Z}_{17}^{6\times 4}$) zufällig gewählt ist, die zweite Matrix ($\mathbb{Z}_{17}^{4\times 1}$) das Geheimnis abbildet, die dritte Matrix ($\mathbb{Z}_{17}^{6\times 1}$) die Unschärfe (bzw. den Fehler) und die vierte Matrix ($\mathbb{Z}_{17}^{6\times 1}$) die Lösung der Gleichung $\mathbf{B} = \mathbf{A} \times \vec{s} + \vec{e}$ darstellt:

$$
\begin{bmatrix}
14 & 15 & 5 & 2 \\
13 & 14 & 14 & 6 \\
6 & 10 & 13 & 1 \\
10 & 4 & 12 & 16 \\
9 & 5 & 9 & 6 \\
3 & 6 & 4 & 5
\end{bmatrix}
\times
\begin{bmatrix}
0 \\ 13 \\ 9 \\ 11
\end{bmatrix}
+
\begin{bmatrix}
1 \\ -1 \\ 0 \\ -1 \\ 1 \\ 0
\end{bmatrix}
=
\begin{bmatrix}
8 \\ 16 \\ 3 \\ 12 \\ 9 \\ 16
\end{bmatrix}
$$

Formal wird das LWE wie folgt definiert [25]:

Gegeben sei die Dimension $n \geq 1$, der Modulus $q \geq 2$ und eine „Fehlerwahrscheinlichkeitsverteilung“ $\chi$ über $\mathbb{Z}_q$. Sei $\mathbf{A}_{\vec{s},\chi}$ über $\mathbb{Z}_q^n \times \mathbb{Z}_q$ die Wahrscheinlichkeitsverteilung, die man erhält, wenn einen Vektor $\vec{a} \in \mathbb{Z}_q^n$ zufällig wählt und Proben der Form $(\vec{a}, \langle \vec{a}, \vec{s} \rangle + e)$ ausgibt, die in $\mathbb{Z}_q$ im Modul $q$ addiert werden, wobei $e \in \mathbb{Z}_q$ gemäß $\chi$ gewählt wird. Gesucht wird ein Algorithmus, der für ein beliebiges $\vec{s} \in \mathbb{Z}_q^n$ bei einer beliebigen Anzahl von unabhängigen Stichproben aus $\mathbf{A}_{\vec{s},\chi}$ das Geheimnis $\vec{s}$ ausgibt.

Die Schwere des LWE ist mit SIS vergleichbar, da auch das LWE auf das SVP reduziert werden kann [29].

#### 2.2.3.3 Learning with Errors over Rings (RLWE)

Das RLWE oder Ring-LWE ist eine Abwandlung des LWE. Beim Ring-LWE Problem wird die Gruppe $\mathbb{Z}_q^n$ durch einen Ring $\mathbb{Z}_q[x]/\langle x^{\mathrm{n}} + 1\rangle$ ersetzt (daher der Name Ring-LWE) [25]. Der Vorteil des Ring-LWE ist die kürzere Schlüssellänge verglichen mit Kryptosystemen, die auf dem SIS oder LWE basieren. SIS und LWE haben typischerweise eine Schlüssellänge der Größe $n^2$, wobei RLWE mit einer linearen Schlüssellänge auskommt [25]. Dementsprechend interessant ist das Ring-LWE für Implementierungen von Kryptosystemen, die in der Praxis angewendet werden.

Die Schwere des Ring-LWE und Ring-SIS sind mit dem SVP vergleichbar, da die Schwere des Ring-SIS mit der des SIS vergleichbar ist [30] und die Schwere des Ring-LWE auf die des Ring-SIS reduziert werden kann [31].

### 2.2.4 Homomorphe gitterbasierte Verschlüsselungsverfahren

In diesem Kapitel werden verschiedene voll-homomorphe Verschlüsselungsverfahren vorgestellt, die auf Gittern basieren.

#### 2.2.4.1 Gentry-Kryptosystem

Das erste in der Literatur bekannte voll-homomorphe Public-Key Kryptosystem wurde von Craig Gentry entwickelt [10]. Als Grundlage benutzt Gentry Gitterideale und einen polynomen Ring. Die Grundidee des Algorithmus basiert auf zwei Schritten, die nacheinander ausgeführt werden, um sowohl Additionen als auch Multiplikationen auf Geheimtexten auszuführen. Der erste Schritt erzeugt ein Rauschen, welches zur Verschleierung dient und die Anzahl an Additionen und Multiplikationen beschränkt, da ab einem bestimmen Rauschniveau die Entschlüsselung nicht mehr korrekt funktioniert. Daher wird im zweiten Schritt ein Bootstrapping durchgeführt, welches das Rauschniveau so absenkt, dass die Geheimtexte korrekt entschlüsselt werden können. Dies funktioniert nur, wenn die homomorphe Funktion die Entschlüsselung mit Hilfe der Evaluierungsfunktion (homomorphe Auswertung) überprüfen kann. Die bahnbrechende Idee von Gentry löst dieses Problem, indem der private Schlüssel in dem Rauschen versteckt wird. Zum besseren Verständnis des

Gentry-Kryptosystem wird die Interpretation des Gentry Algorithmus auf Ganzzahlen [32] im Folgenden vorgestellt, da diese leichter verständlich ist als die ursprüngliche gitterbasierte Version. Dabei unterscheiden sich beide Versionen lediglich bei der Auswahl der Parameter und der homomorphen Basisfunktion [2].

Zunächst wird der erste Schritt erläutert, der noch nicht voll-homomorph ist.

**Schlüsselgenerierung** Der Algorithmus ist zunächst symmetrisch und benötigt daher lediglich einen Schlüssel $p$ mit $p \in (2^{\eta-1}, 2^{\eta}) \cap (2\mathbb{Z}+1)$, wobei $\eta$ die Bitlänge des Schlüssels darstellt.

**Verschlüsselung** Gegeben sind die Zufallszahlen $q, r \in \{-2^{p}, 2^{p}\}$, wobei $p/2 \geqq 2r$. Die Zufallszahl $r$ wird auch als Rauschen bezeichnet. Die Ein-Bit-Nachricht $m \in \{0, 1\}$ wird wie folgt verschlüsselt:

$$c = p * q + 2r + m \tag{2.11}$$

**Formel 2.11:** Verschlüsselung SHE Gentry-Kryptosystem

**Entschlüsselung** Durch $\mod p$ wird der Geheimtext $c$ auf eine Zahl $\in \{-2^{p}, 2^{p}\}$ reduziert, sodass die Entschlüsselung wie folgt berechnet werden kann:

$$m = (c \mod p) \mod 2 \tag{2.12}$$

**Formel 2.12:** Entschlüsselung SHE Gentry-Kryptosystem

**homomorphe Auswertung** Gegeben sind ein Schaltkreis $C$ mit $t$ Eingängen und $t$ Geheimtexten $c_i$. Die Auswertung wird durchgeführt, indem die Ganzzahl Addition- und Multiplikations-Gatter von $C$ auf die Geheimtexte angewendet werden, alle Operationen über den Ganzzahlen durchgeführt werden und schlussendlich das Ergebnis als Ganzzahl zurückgegeben wird.

Um aus dem aktuell symmetrischen Verschlüsslungssystem eine asymmetrisches Verschlüsselungssystem zu generieren, wird die Schlüsselgenerierung und Verschlüsselung erweitert. Der private Schlüssel ist $sk = p$. $\tau$ ist die Anzahl von Ganzzahlen in dem öffentlichen Schlüssel. $\gamma$ ist die Bit-Länge der Ganzzahlen in dem öffentlichen

Schlüssel. $\lambda$ ist der Sicherheitsparameter. $\eta$ ist Bit-Länge des geheimen Schlüssels. Der öffentliche Schlüssel $pk$ ist $pk = (x_0, ..., x_\tau)$ und wird berechnet mit:

$$x_\mathrm{i} = q`_\mathrm{i} * p + 2r_\mathrm{i} \tag{2.13}$$

**Formel 2.13:** Schlüsselerzeugung SHE Gentry-Kryptosystem

Bei der Verschlüsselung wird nun eine Teilmenge $S$ mit $S \subseteq (x_0, ..., x_\tau)$ beliebig gewählt und die Verschlüsselung wie folgt angepasst:

$$c = \sum_{i \in S} x_\mathrm{i} + 2r + m \mod x_0 \tag{2.14}$$

**Formel 2.14:** Verschlüsselung SHE Gentry-Kryptosystem erweitert

Bislang ist der Algorithmus nicht voll-homomorph, da die Addition und Multiplikation nur bis zu einer gewissen Größe des Rauschparameters $r$ funktioniert. Der Algorithmus ist also (bis jetzt) lediglich somewhat homomorph. Daher benötigt es das Bootstrapping, um das Rauschen zu reduzieren. Daher benutzen die Autoren nach dem Vorbild von Gentry eine Transformation, um das Entschlüsselungsgatter zu „zerquetschen“ (eng: squashing). Bei dieser Transformation wird dem öffentlichen Schlüssel zusätzliche Informationen über den geheimen Schlüssel hinzugefügt. Der Geheimtext wird mit dieser zusätzlichen Information bearbeitet. Der bearbeitete Geheimtext kann effizienter entschlüsselt werden als der ursprüngliche Geheimtext, wodurch das Verfahren bootstrapping-fähig wird.

Folgende Schritte müssen modifiziert werden, damit das squashing erfolgreich ist und das Kryptosystem voll-homomorph. Sei die Zufallszahl $K = \gamma * \eta / p`$ und der Rauschparameter $\Theta = w(K \log \lambda)$. Der geheime Schlüssel wird aus dem somewhat homomorphic Kryptosystem übernommen. Der öffentliche Schlüssel wird um die Vektoren $\vec{y} = \{y_1, ..., y_\Theta\}$ ergänzt, sodass es eine Teilmenge $S \subset \{1, ..., \Theta\}$ der Größe $\lambda$ mit $\sum_{i \in S} y_i \approx 1/p \mod 2$ gibt.

**Schlüsselgenerierung** Die überliegenden Schritte müssen ausgeführt werden, damit der geheime und öffentliche Schlüssel wie folgt veröffentlicht werden kann:

$$sk = \vec{s} \quad pk = (pk^*, \vec{y}) \tag{2.15}$$

**Formel 2.15:** Schlüssel FHE Gentry-Kryptosystem

Der Vektor $\vec{s}$ bildet dabei den Indikationsvektor der Teilmenge $S$. $pk^*$ steht für den öffentlichen Schlüssel aus dem somewhat homomorphic Kryptosystem.

**Verschlüsselung und Evaluierung** Der Geheimtext $c^*$ wird wie beim somewhat homomorphic Kryptosystem erzeugt. Mit Hilfe von $i \in \{1, ..., \Theta\}$ wird der Vektor $\vec{z}$ durch $z_i \leftarrow [c^* \cdot y_i]_2$ definiert. Anschließend kann $c^*$ und $\vec{z} = (z_q, ..., z_\Theta)$ veröffentlicht werden.

**Entschlüsselung** Die Entschlüsselung kann wie folgt durchgeführt werden:

$$m` = c - \lfloor \sum_{i=1}^{\Theta} s_\mathrm{i} * u_\mathrm{i} \rceil \mod 2 \tag{2.16}$$

**Formel 2.16:** Entschlüsselung FHE Gentry-Kryptosystem

#### 2.2.4.2 BFV

Das Brakerski-Fan-Vercauteren (BFV) Kryptosystem [33] zählt zu der zweiten Generation der FHE Kryptosysteme [34]. Es baut auf dem Kryptosystem von Brakerski [35] auf, weshalb das Kryptosystem in der Literatur meist BFV und nur selten FV genannt wird. Die Autoren übertragen das Brakerski Kryptosystem von einem LWE zu einem Ring-LWE. BFV wird über zwei Ringe instanziiert. Einen Klartextring, der die Kodierungen der Klartext-Nachrichten enthält, und den Geheimtextring, der die verschlüsselten Nachrichten enthält. Wie jedes andere FHE-Verfahren erlaubt BFV einer nicht vertrauenswürdigen Partei (beispielsweise einer Public Cloud), Berechnungen auf den Geheimtexten durchzuführen, ohne Zugang zum privaten Schlüssel zu haben. Auch hier ist dies auf Grund der homomorphen Eigenschaften möglich. Die Algorithmen zur Schlüsselerzeugung, Verschlüsselung, Entschlüsselung und homomorphen Auswertung können dem Paper [33] entnommen werden.

Verglichen mit dem originalen Kryptosystem von Gentry ist die Effizienz deutlich verbessert. Dies liegt vor allem in der sogenannten „Modulos-Reduktion“, welches das Rauschen reduziert. Die Modulos-Reduktion funktioniert dabei wie folgt[36]:

Gegeben sind $p, q \in (2\mathbb{Z}+1)$, der Vektor $\vec{c}$ und der Vektor $\vec{c'}$, welcher den kürzesten Abstand zu $(p/q) * c$ hat, sodass $c' = c \mod 2$ gilt. Für den privaten Schlüssel $\vec{s}$ mit $|[(\vec{c}, \vec{s})]_q| < q/2 - (q/p) * l_1(\vec{s})$ gilt:

$$[(\vec{c'}, \vec{s})]_p = [(\vec{c}, \vec{s})]_q \mod 2 \quad \text{und} \quad |[(\vec{c'}, \vec{s})]_p| < (p/q) * |[(\vec{c'}, \vec{s})]_q| + l_1(\vec{s}) \tag{2.17}$$

**Formel 2.17:** Modulos-Reduktion

wobei $l_q(\vec{s})$ die Summennorm von $\vec{s}$ ist. Durch diese Methode kann das Rauschen ohne Kenntnis über den privaten Schlüssel und ohne Bootstrapping reduziert werden. Durch das reduzierte Rauschen kann das Kryptosystem als LHE verwendet werden, was die Effizienz verglichen mit dem FHE deutlich steigert.

#### 2.2.4.3 FHEW

Das Fastest Homomorphic Encryption in the West (FHEW) Kryptosystem [37] zählt zu der dritten Generation der FHE Kryptosysteme [34]. Das FHEW erhöht die Effizienz des Bootstrapping deutlich. Auf die genaue Funktionsweise und entsprechenden Algorithmen wird nicht weiter eingegangen, da verschiedene FHE Kryptosysteme bereits vorgestellt wurden. In diesem Abschnitt wird lediglich die Optimierung von FHEW kurz vorgestellt. Grundsätzlich basiert die Effizienzsteigerung auf zwei Hauptfaktoren. Der erste Faktor ist eine neue Technik, um ein NAND-Gatter von zwei LWE-Geheimtexten homomorph zu berechnen. Dazu wird während des Bootstrapping das Modulo von 2 auf 4 erhöht. Dadurch wird das Rauschen reduziert, was die Effizienz erhöht. Der zweite Faktor bezieht sich ebenfalls auf das Bootstrapping. Das Entschlüsseln von LWE-Geheimtexten benötigt die Berechnung eines Skalarprodukts und eine Rundungsoperation. Um die Berechnung möglichst effizient zu gestalten, haben sich die Autoren an dem Algorithmus von Gentry orientiert, der auf Ganzzahlen implementiert ist [32]. Dazu wird ein Ring verwendet, indem jedes Element des Rings in einem Geheimtext kodiert ist und nicht (wie sonst üblich) in einem Vektor aus Geheimtexten.

## 2.3 Mehrparteienberechnungen

In diesem Abschnitt werden die notwendigen Grundlagen zu der Mehrparteienberechnung geschaffen. Dazu wird zunächst die Grundidee erklärt und die verschiede-

nen Verfahren kurz vorgestellt, wobei sich auf die generische Konstruktion fokussiert wird. Diese wird mit Hilfe der Garbled Circuits und einem Beispiel erläutert. Abgeschlossen wird der Abschnitt mit der Erklärung des Oblivious Transfers, welcher für die Garbled Circuits benötigt wird, und dem Ausbau zum quantenresistenten Oblivious Transfer.

### 2.3.1 Grundidee und Verfahren

Die Grundidee der sicheren Mehrparteienberechnung ist, dass verschiedene Parteien gemeinsam eine Berechnung durchführen können, ohne ihre Ein- und Ausgaben offenlegen zu müssen [38]. Dabei sind in der Regel alle Parteien unabhängig voneinander und vertrauen sich gegenseitig nicht. Formal definiert stellt die sichere Mehrparteienberechnung eine sichere Berechnung dar, bei der $m$ Parteien zusammen eine Funktion $f$ mit $f(x_1, x_2, ..., x_{\mathrm{m}})$ auswerten möchten, ohne die Eingabe $x_{\mathrm{i}}$ preisgeben zu müssen. Lediglich das Ergebnis der Funktion $f$ steht allen Parteien zur Verfügung [39]. Diese sichere Berechnung kann über drei verschiedene Implementierungsstrategien zur Verfügung gestellt werden [40]:

1. Generische Konstruktionen durch Garbled Circuits
2. Homomorphe Verschlüsselung
3. Secret Sharing

Eine Unterform der Mehrparteienberechnung ist die Outsourced Computation (ausgelagerte Berechnung). Das besondere bei der Outsourced Computation ist, dass nicht alle Parteien gleichwertig sind, sondern eine Partei der Eigentümer der Daten ist, auf denen Berechnungen ausgeführt werden sollen. Die andere Partei oder Parteien sollen lediglich die Berechnung ausführen, ohne Informationen über die Daten bzw. Eingaben zu gewinnen. Diese Form der Mehrparteienberechnung ist für die Forschungsfrage relevant, weshalb die allgemeine sichere Mehrparteienberechnung nicht weiter betrachtet wird. Für die Outsourced Computation kommen lediglich zwei der drei Implementierungsstrategien in Frage: Generische Konstruktionen durch Garbled Circuits und homomorphe Verschlüsselung. Um diese homomorphe Verschlüsselung für die Outsourced Computation zu nutzen, werden die Daten mit Hilfe eines voll-homomorphen Verschlüsselungssystem verschlüsselt, anschließend für die Berechnungen in Form von Geheimtexten an die andere Partei geschickt und nach der Berechnung auf den Geheimtexten durch den Eigentümer der Daten wieder entschlüsselt. Die homomorphe Verschlüsselung wurde bereits im Abschnitt 2.2.4

vorgestellt. Daher wird nun nur noch die generische Konstruktion durch Garbled Circuits erläutert.

### 2.3.2 Generische Konstruktion

Yao stellte seinen generischen Ansatz 1982 vor, indem er das Millionärsproblem mit Hilfe von Garbled Circuits löste. Bei dem Millionärsproblem möchten zwei Millionäre herausfinden, welcher von beiden mehr Geld besitzt, ohne dem jeweils anderen zu verraten wie hoch das eigene Vermögen ist [41]. Das vorgestellte Protokoll namens Garbled Circuits (deutsch: verkrüppelte bzw. verschleierte Schaltkreise) basiert auf einem Schaltkreis, welcher es zwei Personen ermöglicht die gewünschte Berechnung auszuführen, ohne die Eingabe preisgeben zu müssen. Da sich jeder Algorithmus mit verteilten Eingabewerten mit Hilfe von Garbled Circuits darstellen lässt [42], heißt dieser Ansatz auch generischer Ansatz bzw. generische Konstruktion.

Das Garbled Circuit Protokoll besteht aus zwei Schritten: Der Konstruktion und der Berechnung. Das Protokoll ist zur Veranschaulichung in Abbildung 8 abgebildet.

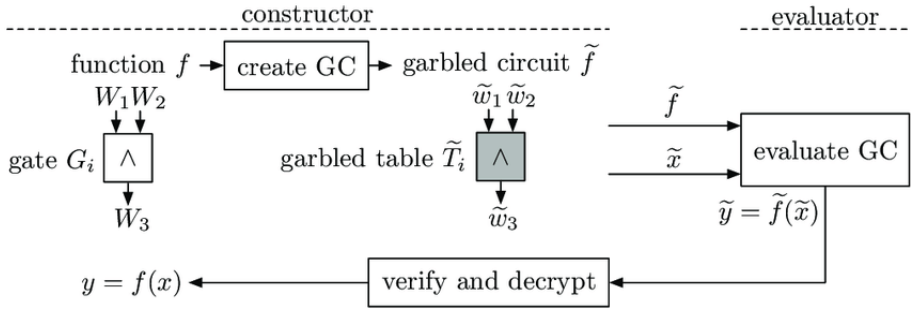



**Abbildung 8:** Abbildung des Garbled Circuits Protokoll (Quelle: [43])

**Konstruktion des Garbled Circuit** Der Schaltkreis der Funktion $f$ wird von dem Constructor verschlüsselt. Dazu wird mit Hilfe der Wahrheitstabelle des Schaltkreises eine verschleierte Wahrheitstabelle $\tilde{T}_{\mathrm{i}}$ erstellt. Die verschleierte Wahrheitstabelle wird mit zufälligen Zeichenketten (Labels) gefüllt, die die eigentlichen Werte der Eingänge $W_1, W_2$ in Form von $0, 1$ repräsentieren, jedoch keinen Rückschluss auf die einzelnen Zustände der Eingänge lassen. Beim Entschlüsseln können nur die Zustände der Ausgänge ermittelt werden [41].

Das folgende Beispiel erklärt den Aufbau eines Garbled Circuit am Beispiel eines Halbaddierer-Schaltkreises. Die Abbildung 9 zeigt den Schaltkreis.

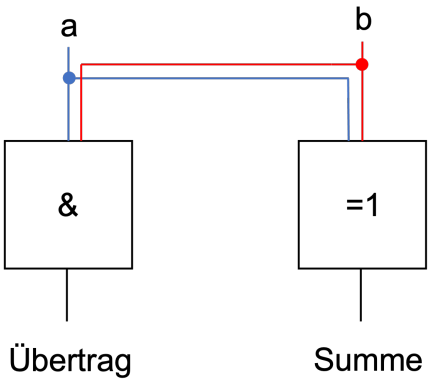



**Abbildung 9:** Abbildung eines Halbaddierer-Schaltkreis (Quelle: Eigene Zeichnung angelehnt an [44])

Der Halbaddierer besteht aus zwei Gattern, die jeweils zwei Eingänge und einen Ausgang besitzen. Die Zustände der Gatter können über Wahrheitstabellen abgebildet werden. Die Tabelle 2 zeigt dies exemplarisch für ein XOR-Gatter.

| a | b | Ausgang |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**Tabelle 2:** Wahrheitstabelle für ein XOR-Gatter

In dem Beispiel ist Alice der Constructor und Bob der Evaluator. Beide möchten die Funktion $f(a, b)$ berechnen, ohne dem jeweils anderen preiszugeben, welchen Wert die eigene Eingabe hat. In diesem Beispiel möchten Alice und Bob ihre beiden Zahlen addieren, ohne, dass der andere erfährt, um welche Zahl es sich handelt[1]. Alice wählt $a$ mit 1 und Bob wählt $b$ mit 0. Die verschleierte Wahrheitstabelle wird nun wie folgt aufgebaut. Die Wahrheitstabelle wird um zufällige Zeichenketten, also die Labels, erweitert. Wie die Labels generiert werden, spielt keine Rolle, es darf nur von den Labels nicht möglich sein auf die eigentliche Wahrheitstabelle zu schließen. Zur besseren Lesbarkeit bestehen die Labels in diesem Beispiel aus Ganzzahlen mit drei

[1] In diesem Beispiel ist es trivial die Zahl des Anderen herauszufinden, indem Alice und Bob im Nachhinein ihre Zahl von der Summe subtrahieren. Dies wird nicht weiter beachtet, da es lediglich um die Veranschaulichung der Funktionsweise des Garbled Circuit Protokoll geht.

Ziffern. Alice wählt für Ihre Eingaben $(0, 1) = (810, 831)$ und für die Eingaben von Bob $(0, 1) = (170, 820)$. Diese Information, also welches Label zu welcher Eingabe gehört, stellt Alice Bob nicht zur Verfügung. Die Verschlüsselung für das XOR-Gatter wird in Tabelle 3 abgebildet. Dabei wird ein einfacher Verschlüsselungsalgorithmus verwendet: $enc(output, input1, input2) = output + hash(index, input1, input2)$. Die Input-Parameter sind die zuvor gewählten Labels. Der verschlüsselte Ausgang (dritte Spalte in Tabelle 3) wird an Bob geschickt.

| **a** | **b** | **Ausgang** | **Verschlüsselung** |
|---|---|---|---|
| 0 | 0 | 0 | 0 + hash(2,810,170) |
| 0 | 1 | 1 | 1 + hash(2,810,820) |
| 1 | 0 | 1 | 1 + hash(2,831,170) |
| 1 | 1 | 0 | 0 + hash(2,831,820) |

**Tabelle 3:** Wahrheitstabelle für ein XOR-Gatter mit Verschlüsselungsalgorithmus

**Berechnung** Die Berechnung der Funktion führt der Evaluator aus. Dazu nutzt der Evaluator die verschleierte Wahrheitstabelle von jedem Gatter und entschlüsselt den Geheimtext, der die realen Eingaben des Gatters repräsentiert. Allerdings weiß der Evaluator nicht, welches der passende Geheimtext ist, da bei der Konstruktion der verschleierten Wahrheitstabelle die Labels zufällig gewählt wurden. Um beim Austausch nichts über die eigenen Eingaben zu verraten, bedarf es dem Oblivious Transfer. Dieser ermöglicht es den Austausch zu ermöglichen, ohne die Eingabe zu verraten. Wie der Oblivious Transfer funktioniert, wird weiter unten erklärt. Nachdem die Berechnung abgeschlossen ist, wird die Ausgabe der Funktion in Form einer verschlüsselten Zeichenkette an den Constructor übermittelt, welcher die Entschlüsselung vornimmt und somit das Ergebnis erhält.

Die Berechnung des Beispiels von oben funktioniert nun wie folgt: Bob hat von Alice für jedes Gatter vier Nachrichten als verschleierte Wahrheitstabelle bekommen. Zusätzlich bekommt Bob die Eingabe von Alice als Label (831 in diesem Fall). Mit Hilfe des Oblivious Transfer und den zur Verfügung gestellten Informationen arbeitet Bob ein Gatter nach dem anderen ab. Bob beginnt bei dem AND-Gatter und kennt Alice Eingabe als zufällige Zeichenkette (831), seine eigene Zahl (0) und die verschleierte Wahrheitstabelle des AND-Gatter, welche aus der Tabelle 4 abzulesen ist. Um die Ausgabe des Gatters berechnen zu können, ermittelt Bob mit Hilfe

des Oblivious Transfer, dass die zufällige Zeichenkette, die zu seiner Eingabe passt, 170 ist. Alice erfährt dabei nicht, dass die Eingabe von Bob 0 ist und Bob erfährt nicht, wie das andere Label (820) lautet. Bob nutzt anschließend die Verschlüsselungsfunktion mit den Ihm bekannten Werten, um die Ausgabe (0) zu bestimmen. Um die Ausgabe bestimmen zu können, muss Bob wissen, wie der Ausgang mit der Verschlüsselung verknüpft wird. In diesem Beispiel durch eine Addition mit dem Ergebnis der Verschlüsselungsfunktion.

| **a** | **b** | **Ausgang** | **Verschlüsselung** |
|---|---|---|---|
| 0 | 0 | 0 | 0 + hash(1,810,170) |
| 0 | 1 | 0 | 0 + hash(1,810,820) |
| 1 | 0 | 0 | 0 + hash(1,831,170) |
| 1 | 1 | 1 | 1 + hash(1,831,820) |

**Tabelle 4:** Wahrheitstabelle für ein AND-Gatter mit Verschlüsselungsalgorithmus

### 2.3.3 Oblivious Transfer

Der Oblivious Transfer (OT) ist ein kryptografisches Primitiv, welches für das Garbled Circuits Protokoll benötigt wird. Dabei wird die einfachste Form des OT namens 1-out-of-2 Oblivious Transfer verwendet. Der Sender hat zwei Eingabenachrichten $m_0, m_1$ und der Empfänger ein Auswahl-Bit $c$. Durch das OT Protokoll bekommt der Empfänger die Nachricht $m_\mathrm{c}$, während der Sender keine neuen Informationen erhält [46]. Im Falle der Garbled Circuits kann also Bob die passende Zeichenkette zu seiner Eingabe erhalten, ohne dass Alice erfährt, wie seine Eingabe ist.

Es gibt verschiedene Implementierungen des OT Protokolls. Zum Verständnis wird in Abbildung 10 ein OT Protokoll entwickelt von [45] vorgestellt, welches eine Abwandlung des Diffie-Hellmann Key Exchange (DHKE) darstellt. Dabei wird vorausgesetzt, dass dem Leser der Diffie-Hellmann Key Exchange vertraut ist[2]. In diesem Fall ist Alice der Sender und Bob der Empfänger. Die erste Nachricht ist identisch mit dem DHKE, allerdings berechnet Bob nun $B$ in Abhängigkeit von $c$. Ist $c = 0$, dann berechnet Bob $B = g^\mathrm{b}$. Ist $c = 1$, dann berechnet Bob $B = Ag^\mathrm{b}$. Anschließend leitet Alice zwei Schlüssel $k_0, k_1$ aus $(B)^a$ bzw. $(B/A)^a$ ab. Bob kann jetzt

[2] Als Hilfestellung ist der Diffie-Hellmann Key Exchange (DHKE) ebenfalls in Abbildung 10 abgebildet.

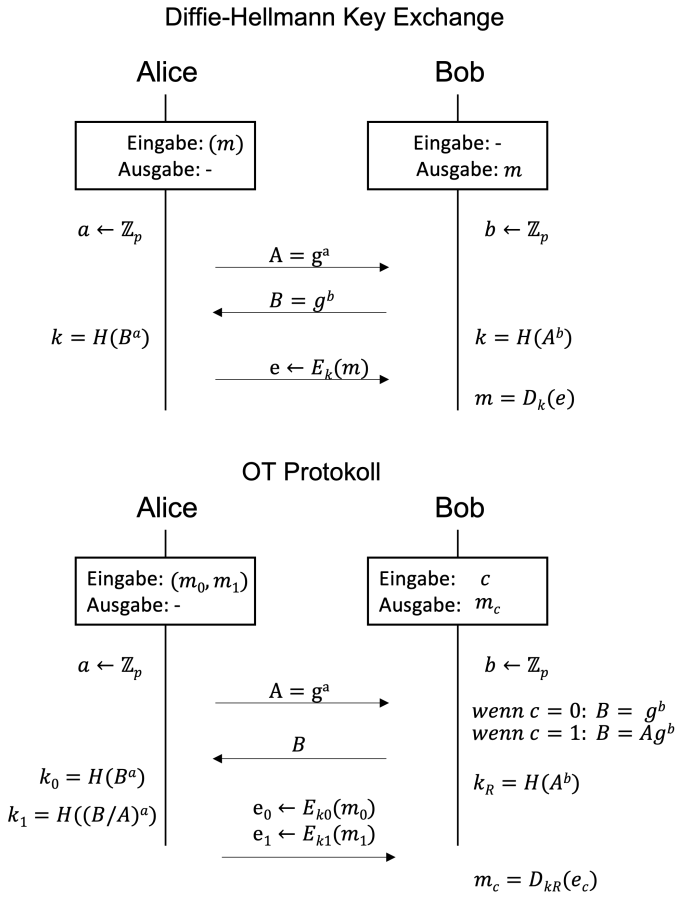



**Abbildung 10:** Abbildung des DHKE und eines OT Protokoll (Quelle: Eigene Zeichnung angelehnt an [45])

den Schlüssel $k_c$ ableiten, der dem von Ihm gewählten Bit aus $A^b$ entspricht. Den anderen allerdings nicht.

### 2.3.4 Post-Quantum Oblivious Transfer

Da der Oblivious Transfer (OT) ein Protokoll ist, kann das unterliegende Kryptosystem nach Belieben ausgetauscht werden. Für diese Arbeit es unabdingbar, dass der Oblivious Transfer quantenresistent ist, weshalb der OT diesbezüglich angepasst werden muss. Im Folgenden wird ein Protokoll [47] vorgestellt, welches das BFV Kryptosystem als Grundlage verwendet. Das Protokoll ist in Abbildung 11 abgebildet. Das OT Protokoll ist in zwei Phasen, der Setup und der OT Phase, unterteilt. Die Setup Phase muss zuerst und nur einmal pro Sitzung ausgeführt werden. Das Besondere an dem Protokoll ist der Fokus auf eine Optimierung hinsichtlich der gesamten Durchlaufzeit. So kann das Protokoll mit $\gamma$ Eingaben gleich-

zeitig umgehen. In der Abbildung 11 wurde auf Grund der Übersichtlichkeit auf die Notation mit $\gamma$ verzichtet. So wäre beispielsweise die Eingabe von Bob eigentlich $c = (c_i)_{i \in [\gamma]} \in (0,1)^{\gamma}$. Zum Verständnis des Protokolls genügt die Darstellung ohne $\gamma$. Es können $n$ OTs parallel ausgeführt werden, indem ein Klartext gewählt wird, welcher mehr als $l$ Bits enthält. Der Sender kann dementsprechend $n$ Eingaben in einen Klartext packen und der Empfänger kann $n$ Auswahl Bits auf einmal verschlüsseln. Zu diesem Zweck stellt das Protokoll drei Funktionen $PackM()$, $UnpackM()$ und $PackC()$ bereit. Auch, wenn auf die $\gamma$ Notation verzichtet wurde, werden die Parallelisierungsfunktionen in der Abbildung aufgezeigt, da sonst das Verständnis diesbezüglich fehlen würde.

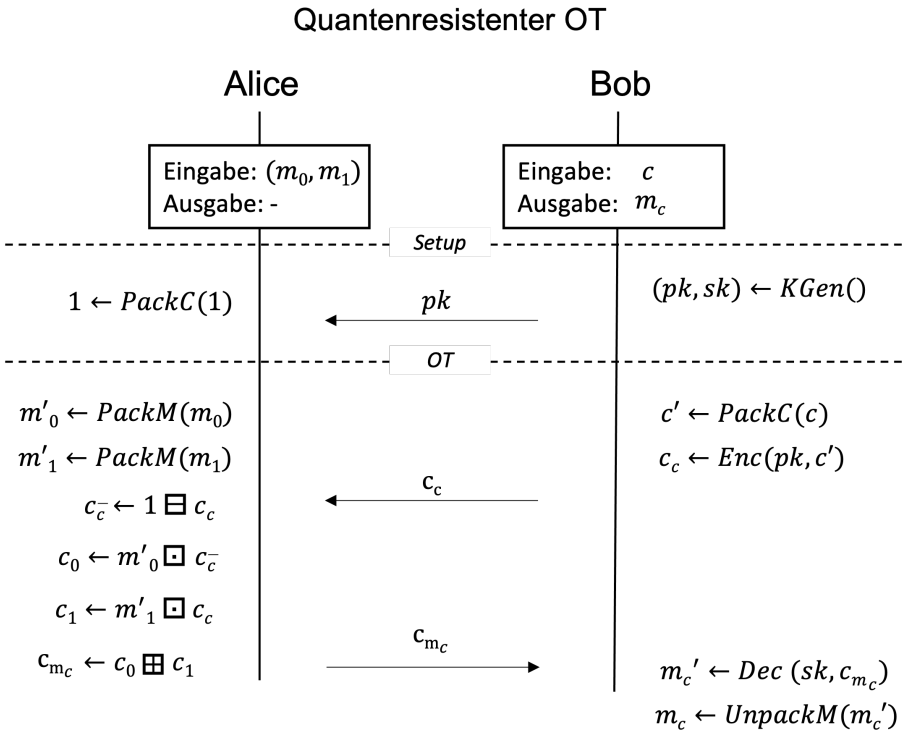



**Abbildung 11:** Abbildung eines quantenresistenten OT Protokolls (Quelle: Eigene Zeichnung angelehnt an [47])

### 2.3.5 Optimierung der Garbled Circuits

Über die Jahre wurde das GC Protokoll immer weiter optimiert. Die Tabelle 5 gibt eine Übersicht der bekannten Optimierungen an den Garbled Circuits. Auf die Funktionsweise der jeweiligen Optimierungsschritte wird auf Grund der fehlenden Relevanz für die Forschungsfrage nicht weiter eingegangen. Allerdings wird sich in

| | **Anzahl der Geheimtexte** | | **Nachrichten pro Schaltung** | | | |
|---|---|---|---|---|---|---|
| | | | Generator | | Evaluator | |
| **Protokoll** | XOR | AND | XOR | AND | XOR | AND |
| klassisch [41] | 4 | 4 | 4 | 4 | 4 | 4 |
| point & permute [48] | 4 | 4 | 4 | 4 | 1 | 1 |
| row reduction [49] | 2 | 2 | 4 | 4 | 1 | 1 |
| free XOR [50] + row reduction [51] | 0 | 3 | 0 | 4 | 0 | 1 |
| fleXOR [52] | {0,1,2} | 2 | {0,2,4} | 4 | {0,1,2} | 1 |
| half gates [53] | 0 | 2 | 0 | 4 | 0 | 2 |

**Tabelle 5:** Übersicht der Optimierungen der Garbled Circuits. (Quelle: Direkt übernommen aus [53]

Kapitel 3.3 (Grenzen und Ausblick beider Prototypen) auf die Optimierungen bezogen, weshalb diese hier kurz aufgeführt werden.

## 2.4 Cloud-Computing

In diesem Abschnitt werden die Grundlagen des Cloud-Computings erläutert. Dazu gehören die verschiedenen Dienstmodelle, Architekturen und Technologien, um ein Verständnis für den Betrieb der Prototypen in der Cloud zu bekommen. Abgeschlossen wird der Abschnitt mit einer Bewertung hinsichtlich der Vertraulichkeit der Daten und der verschiedenen Architekturen.

### 2.4.1 Grundidee

Die Grundidee des Cloud-Computing ist ein flexibles Bereitstellungsmodell von Ressourcen und Diensten über ein Netzwerk (z.B. Internet) [54]. Die dafür notwendige

IT wird vom Cloud-Anbieter dem Cloud-Nutzer[3] zur Verfügung gestellt. Cloud-Computing besitzt fünf essentielle Eigenschaften [55]:

1. Selbstverwaltung auf Abruf
2. umfassende Erreichbarkeit über ein Netzwerk
3. Bündelung der Ressourcen
4. schnelle Flexibilität bezüglich der Verteilung der Ressourcen
5. genaue Abrechnung der in Anspruch genommenen Dienstleistung

Aus diesen Eigenschaften lassen sich die Vorteile der Cloud-Nutzung gegenüber einer On-Premise-Lösung ableiten. Die Cloud[4] kann jederzeit, von überall auf der Welt, kurzfristig, eine dynamisch skalierende Anzahl von Ressourcen zur Verfügung stellen, die genau nach genutzter Leistung abgerechnet werden kann. Somit eignet sich die Cloud vor allem dann, wenn man kurzfristig oder unregelmäßig eine hohe Rechenleistung benötigt.

### 2.4.2 Technologien

Um die folgenden Definitionen, Erklärungen und Abbildungen zu verstehen, werden wichtige Technologien, die beim Cloud-Computing verwendet werden, erläutert.

**Virtualisierung** Die Virtualisierungstechnologie wird im Cloud-Computing vor allem für die Virtualisierung der Betriebssysteme genutzt. Es ermöglicht die logische Trennung von physikalischen Ressourcen. Somit können mehrere Betriebssysteme dieselbe Hardware parallel nutzen.

**Hypervisor** Ein Hypervisor in einer Cloud dient zur Verteilung der physischen Ressourcen. Dabei verwaltet der Hypervisor die physische Hardware und verteilt diese entsprechend der Notwendigkeit auf verschiedene virtuelle Maschinen.

[3]Eine Unterscheidung zwischen Cloud und Service Nutzer findet nicht statt, da dies für die Beantwortung der Forschungsfrage nicht relevant ist.
[4]Der Begriff *Cloud* wird in dieser Arbeit synonym mit dem Begriff Cloud-Computing verwendet.

### 2.4.3 Dienstmodelle

Im Cloud-Computing gibt es drei verschiedene Dienstmodelle, die sich nach der Tiefe vom Cloud-Nutzer zu verwaltenden Schichten unterscheiden. Dabei übernimmt der Cloud-Anbieter in allen Dienstmodellen immer die unteren Schichten, wie in Abbildung 12 erkennbar. Würde der Cloud-Anbieter die unteren Schichten nicht verwalten, dann würde es nicht mehr um eine Cloud-, sondern um eine On-Premise-Lösung halten. Grundsätzlich wird zwischen den folgenden Dienstmodellen unterschieden [55].

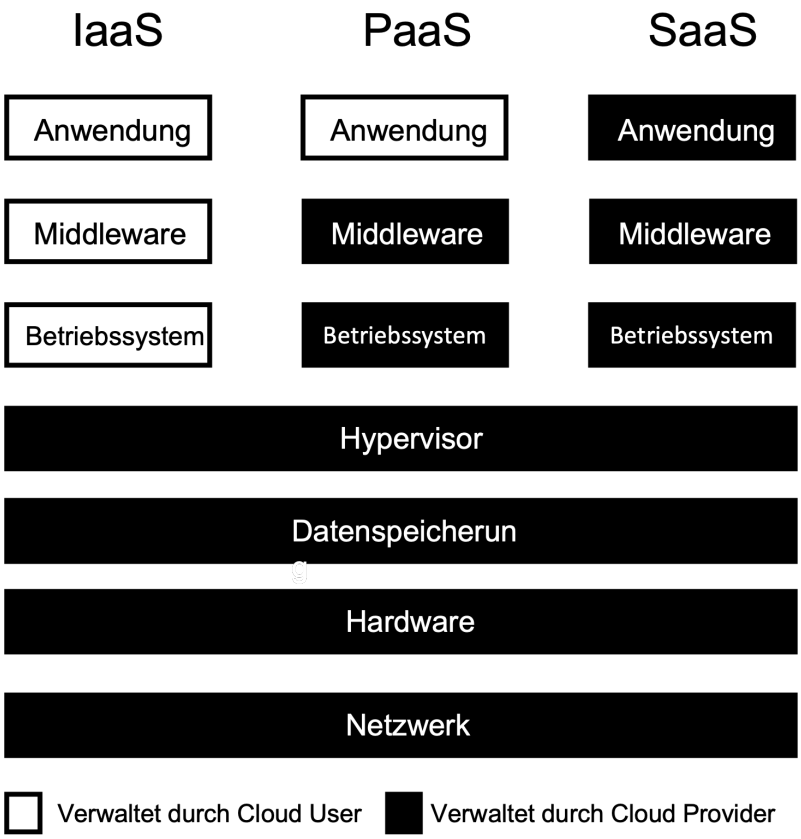



**Abbildung 12:** Gegenüberstellung Dienstmodelle Cloud-Computing (Quelle: Eigene Zeichnung angelehnt an [56])

**Infrastructure as a Service** Das Modell Infrastructure as a Service (IaaS) stellt dem Cloud-Nutzer Hardware zur Verfügung. Dabei handelt es sich in der Regel nicht um dedizierte Hardware, sondern um eine virtuelle Maschine, damit die Ressourcen flexibel zwischen verschiedenen Cloud-Nutzern verteilt werden bzw. abgeschaltet

werden können. Der Cloud-Nutzer kann frei über das Betriebssystem und alle darüber liegenden Schichten verfügen.

**Plattform as a Service** Das Modell Plattform as a Service (PaaS) stellt dem Cloud-Nutzer einen Anwendungs-Server zur Verfügung. Auf diesem Server kann der Cloud-Nutzer seine Anwendung programmieren bzw. installieren. Der Cloud-Anbieter verwaltet alle darunterliegenden Schichten und kann somit die Ressourcen flexibel verwalten bzw. verteilen.

**Software as a Service** Das Modell Software as a Service (SaaS) stellt dem Cloud-Nutzer die fertige Anwendung zur Verfügung. Der Cloud-Nutzer kann die Anwendung nutzen, allerdings nicht weiterentwickeln bzw. auf dem Server modifizieren. Der Cloud-Anbieter verwaltet alle darunterliegenden Schichten und kann somit die Ressourcen flexibel verwalten bzw. verteilen.

### 2.4.4 Cloud-Architekturen

Die verschiedenen Dienstmodelle können über verschiedene Architekturen angeboten werden. Grundsätzlich unterscheidet man zwischen drei[5] verschiedenen Cloud-Architekturen[55].

**Public Cloud** Bei der public Cloud wird das Cloud-Computing über einen externen Cloud-Anbieter ermöglicht. Die Cloud wird der Öffentlichkeit (daher der Name public) angeboten. In diesem Modell können Skaleneffekte besonders gut genutzt werden, da oftmals viele Kunden die public Cloud parallel nutzen und somit eine effiziente und effektive Verteilung der Ressourcen möglich ist.

**Private Cloud** Bei der private Cloud wird das Cloud-Computing durch einen internen Cloud-Anbieter ermöglicht. Beispielsweise durch eine Abteilung in einem Unternehmen, wobei die private Cloud nicht der Öffentlichkeit, sondern nur Internen zur Verfügung gestellt wird (daher der Name private). In diesem Modell hängt es von der Anzahl der Nutzer ab, ob die Skaleneffekte ähnlich gut wie bei der Public Cloud genutzt werden können. In der Regel ist dem nicht so.

[5] Die Community Cloud, die teilweise als viertes Modell aufgeführt wird, wird auf Grund der fehlenden Relevanz für die Beantwortung der Forschungsfrage vernachlässigt.

**Hybride Cloud** Bei der hybriden Cloud wird sowohl eine public als auch eine private Cloud benutzt. In diesem Modell werden sowohl Vor-als auch Nachteile aus beiden Architekturen vereint.

### 2.4.5 Vertraulichkeit der Daten

Die Vertraulichkeit der Daten hängt mit der gewählten Architektur zusammen.

Bei der public Cloud kann der Cloud-Anbieter theoretisch auf alle Daten zugreifen, da dieser die Hardware, auf der alle Daten verarbeitet bzw. gespeichert werden, verwaltet. Dabei spielt es keine Rolle, welches Dienstmodell benutzt wird. Die Vertraulichkeit der Nutzer- und Programmdaten ist bei der public Cloud nicht sichergestellt.

Bei der privaten Cloud kann der Cloud-Anbieter wie bei der public Cloud auf alle Daten zugreifen. Allerdings ist der Cloud-Anbieter bei einer privaten Cloud in der Regel vertrauenswürdig[6]. Dies gilt für einen (kommerziellen) public Cloud-Anbieter nicht. Somit ist die Vertraulichkeit der Nutzer- und Programmdaten in der privaten Cloud sichergestellt.

Bei der hybriden Cloud kommt es auf das Modell des Betriebs an. Die Vertraulichkeit der Daten hängt mit der Fragestellung zusammen, wofür die public Cloud benutzt wird. Dient die public Cloud als eine Art Back-Up für ein Hochverfügbarkeitssystem, dann ist die Einschätzung hinsichtlich der Vertraulichkeit dem der public Cloud gleichzusetzen. Dient die public Cloud als Berechnungsauslagerung von Prozessen, die keine Anforderung an die Vertraulichkeit haben, dann ist die Einschätzung hinsichtlich der Vertraulichkeit dem der privaten Cloud gleichzusetzen.

## 2.5 Quantenresistente vertrauliche Cloud-Nutzung

In diesem Abschnitt werden die beiden quantenresistenten Möglichkeiten der Cloud-Nutzung aus der Forschungsfrage vorgestellt und miteinander verglichen. Der Fokus liegt dabei auf der Quantenresistenz und der Vertraulichkeit beider Verfahren.

[6]Theoretisch könnte ein Mitarbeiter bzw. Interner einen Angreifer darstellen. Dieses Szenario wird in dieser Arbeit nicht berücksichtigt.

### 2.5.1 Homomorphe Verschlüsselung durch gitterbasierte Verfahren

Die Grundidee der homomorphen Nutzung der Cloud ist in der Abbildung 13 erkennbar. Der Cloud-Nutzer verschlüsselt die Klartexte vor der Verwendung der Cloud mit einem homomorphen Verschlüsselungsverfahren. Nachdem die Klartexte verschlüsselt sind, werden die Geheimtexte an die Cloud verschickt. Die gewünschten Berechnungen finden in der Cloud auf den Geheimtexten statt und werden anschließend wieder an den Cloud-Nutzer geschickt. Dieser entschlüsselt die Geheimtexte außerhalb der Cloud und erhält so das Ergebnis der gewünschten Berechnung.

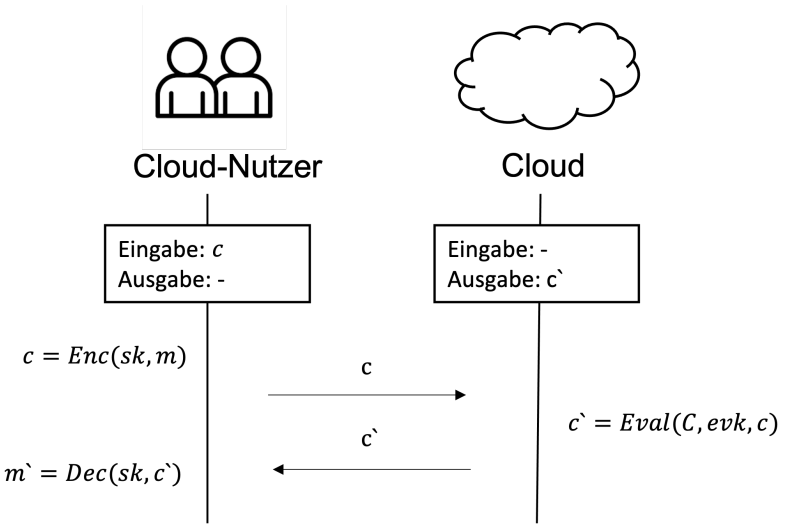



**Abbildung 13:** Homomorphe Nutzung der Cloud (Quelle: Eigene Zeichnung)

Die Vertraulichkeit der Daten ist auf Grund der Verschlüsselung sichergestellt. Wie sicher die Verschlüsselung ist, hängt von dem ausgewählten Verfahren ab. Die Testläufe mit dem Prototypen, welcher in Kapitel 3.1 vorgestellt wird, wurden mit einer 128-Bit Verschlüsselung durchgeführt.

Die Quantenresistenz hängt ebenfalls von dem ausgewählten Verfahren ab. Grundsätzlich gilt die gitterbasierte Kryptographie und somit die verwendeten Verfahren als quantenresistent ([57]). Die verwendeten Verfahren des Prototyps und die Gründe für die Auswahl werden in Kapitel 3.1 erläutert.

### 2.5.2 Mehrparteienberechnung

Um die Mehrparteienberechnung für die vertrauliche Cloud-Nutzung verwenden zu können, basiert der in Kapitel 3.2 vorgestellte Prototyp auf dem Twin-Cloud-Modell

[58], welches nun kurz und in Kapitel 3.2 ausführlich dargestellt wird. Die Abbildung 14 zeigt den Aufbau des Modells. Dabei werden zwei Clouds, eine private Cloud und eine public Cloud benötigt. Es kann somit auch von einer hybriden Cloud-Nutzung gesprochen werden. Der Cloud-Nutzer schickt die Klartexte an die private Cloud, welche als Constructur in dem Garbled Circuits Protokoll fungiert. Die public Cloud bildet den Evaluator aus dem Garbled Circuits Protokoll ab. Das Garbled Circuits Protokoll wird vollständig ausgeführt und das Ergebnis der Berechnung wird nur der privaten Cloud und dem Cloud-Nutzer zur Verfügung gestellt.

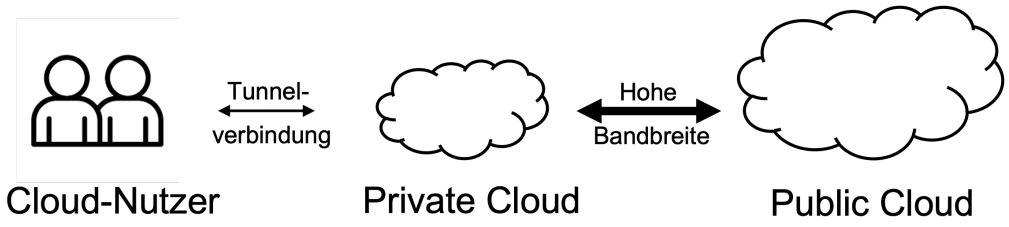



**Abbildung 14:** Aufbau Twin-Cloud-Modell (Quelle: Eigene Zeichnung angelehnt an [58])

Die Vertraulichkeit der Daten ist auf Grund der Verschleierung sichergestellt, sofern das Garbled Circuits Protokoll mit sicheren Verfahren ausgeführt wird. Die Testläufe mit dem Prototypen, welcher in Kapitel 3.1 vorgestellt wird, wurden mit einer 256-Bit AES-Verschlüsselung für die Verschleierung der Wahrheitstabellen und einer 128-Bit BFV-Verschlüsselung für den OT durchgeführt.

Die Quantenresistenz hängt ebenfalls von dem ausgewählten Verfahren ab. Grundsätzlich gelten symmetrische Verfahren als quantenresistent, sofern die Schlüssellänge verdoppelt wird ([59]). Aus diesem Grund wird AES mit 256-Bit Schlüsseln im Prototypen genutzt, um auf eine 128-Bit Sicherheit zu kommen. Das OT Protokoll nutzt als Grundlage BFV und ist daher quantenresistent. Der ausführliche Beweis ist in [47] aufgeführt. Die begründete Auswahl der verwendeten Verfahren ist in Kapitel 3.2 beschrieben.

### 2.5.3 Vergleich von homomorpher Verschlüsselung und Mehrparteienberechnung

Die Tabelle 6 zeigt einen Vergleich beider Verfahren. Beide Verfahren erfüllen die gewünschten Anforderungen an Sicherheit, Quantenresistenz und Vertraulichkeit.

Der größte Unterschied ist, dass die homomorphe Verschlüsselung mit der Rechenleistung skaliert und die Mehrparteienberechnung mit der Netzwerkverbindung.

| **Parameter** | **Homomorphe Verschlüsselung** | **Mehrparteien -berechnung** |
|---|---|---|
| Geschwindigkeit | basiert auf Rechenleistung | basiert auf Netzwerkverbindung |
| Interaktivität | nicht gegeben | gegeben |
| Quantenresistenz | gegeben | gegeben |
| Vertraulichkeit | gegeben | gegeben |
| Sicherheit | min. 128-Bit gegeben | min. 128-Bit gegeben |

**Tabelle 6:** Vergleich der homorphen Verschlüsselung und Mehrparteienberechnung

# 3 Architektur zur quantenresistenten vertraulichen Cloud-Nutzung

In diesem Kapitel werden die Prototypen beider Verfahren vorgestellt. Der Fokus liegt hierbei auf der Erläuterung der Architektur und weniger auf der konkreten Implementierung. Zudem werden die Grenzen und Optimierungsmöglichkeiten beider Prototypen kritisch beleuchtet. Das Kapitel wird mit den Bewertungskriterien abgeschlossen, die für die Evaluation in Kapitel 4 benötigt werden.

## 3.1 Homomorphe Verschlüsselung

In diesem Abschnitt wird der Prototyp vorgestellt, der auf der homomorphen Verschlüsselung basiert.

### 3.1.1 Architektur

Der Prototyp basiert auf der Implementierung einer Krypto-Bibliothek, welche in Abschnitt 3.1.2 genauer beschrieben wird. Es werden verschiedene Klassen homomorpher Verschlüsselung unterstützt, da der Aufbau über verschiedene Schichten (ähnlich dem ISO OSI-Referenzmodell) abgebildet wird. Die Abbildung 15 zeigt die verschiedenen Schichten .

Dieses Schichtenmodell ermöglicht einen flexiblen Einsatz für die Evaluation, da sich die verwendeten Verfahren in jeder Schicht leicht austauschen lassen. Die unterste Schicht stellt die primitiven mathematischen Grundlagen, wie z.B. Transformationen oder Modulo bereit. Diese Operationen können parallel und hardwareoptimiert ausgeführt werden. Die nächste Schicht stellt die notwendigen Gitterfunktionen bereit. Dazu zählen beispielsweise Gitter und Ringe generell, aber auch die grundlegenden Operationen auf beiden. Die nächsthöhere Schicht ist die Krypto-Schicht. Diese Schicht bildet die homomorphen Verschlüsselungsverfahren bzw. die eigentlichen Public-Key-Protokolle ab. Die oberste Schicht ist die Anwendungsschicht, welche die Tests für die Evaluierung und die Dekodierung enthält.

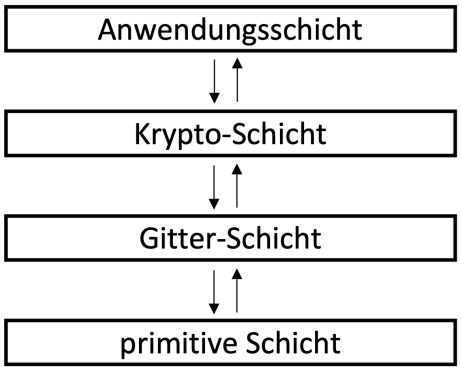



**Abbildung 15:** Homomorphe Nutzung der Cloud (Quelle: Eigene Zeichnung angelehnt an [60]

Es werden zwei Klassen der homomorphen Verschlüsselung unterstützt: Die somewhat homomorphe Verschlüsselung und die voll-homomorphe Verschlüsselung. Für die Berechnungen auf den Geheimtexten gibt es drei Varianten.

1. exakte Arithmetik
2. angenäherte Arithmetik
3. Schaltungen (bzw. Gatter)

Die beiden Arithmetik-Varianten sind für die Evaluation lediglich bedingt relevant. Auf Grund der in Abschnitt 3.3 geschilderten Bewertungskriterien eignet sich die gatterbasierte Variante am besten, da genau die gleichen Schaltungen wie bei der Mehrparteienberechnung zur Berechnung benutzt werden können. Allerdings wird eine Arithmetik-Variante für den OT bei der Mehrparteienberechnung benötigt, weshalb beide Varianten kurz und die gatterbasierte Variante ausführlicher vorgestellt werden.

#### 3.1.1.1 Exakte Arithmetik

Die Tabelle 7 gibt einen Überblick über die Eigenschaften der exakten Arithmetik-Variante.

Die exakte Arithmetik-Variante kann SIMD-Operationen effizient ausführen, indem mit mehreren Vektoren aus Ganzzahlen parallel in einem Durchlauf gerechnet wird. Die grundlegenden arithmetischen Funktionen lassen sich schnell berechnen, allerdings benötigt das Bootstrapping viele Ressourcen und ist dementsprechend langsam, sodass das Bootstrapping möglichst vermieden werden sollte.

| Eigenschaft | Ausprägung |
|---|---|
| Klartext | Vektoren |
| Berechnung | Ganzzahl Arithmetik |
| Kryptosysteme | BGV, BFV |
| Klasse | SHE |

**Tabelle 7:** Eigenschaften der exakten Arithmetik-Variante

#### 3.1.1.2 Angenäherte Arithmetik

Die Tabelle 8 gibt einen Überblick über die Eigenschaften der angenäherten Arithmetik-Variante.

| Eigenschaft | Ausprägung |
|---|---|
| Klartext | Gleitkommazahlen |
| Berechnung | Gleitkommazahlen Arithmetik |
| Kryptosysteme | CKKS |
| Klasse | SHE |

**Tabelle 8:** Eigenschaften der angenäherten Arithmetik-Variante

Die angenäherte Arithmetik gleicht der exakten Arithmetik, allerdings werden Gleitkommazahlen unterstützt. Auch bei dieser Variante werden SIMD-Operationen unterstützt, allerdings sollte aus Performancegründen auf das Bootstrapping verzichtet werden.

#### 3.1.1.3 Gatterbasiert

Die Tabelle 9 gibt einen Überblick über die Eigenschaften der gatterbasierten Variante.

Die gatterbasierte Variante unterstützt sämtliche Schaltungen, weshalb sich diese Variante besonders gut für den Vergleich mit der Mehrparteienberechnung eignet, da diese auf Grund des Garbled Circuits Protokolls auch mit Schaltungen arbeitet. Ein weiterer Vorteil der gatterbasierten Variante ist die hohe Schnelligkeit, da das Bootstrapping verglichen mit den anderen Varianten deutlich schneller ist. Daher

| Eigenschaft | Ausprägung |
|---|---|
| Klartext | Bits |
| Berechnung | Schaltungen |
| Kryptosysteme | FHEW, TFHE |
| Klasse | FHE |

**Tabelle 9:** Eigenschaften der gatterbasierten Variante

ist die voll-homomorphe Verschlüsselung möglich, ohne, dass es wie bei den anderen Varianten zu (nicht akzeptierbaren) langen Laufzeiten kommt. Die Schaltungen können beliebig groß sein, ohne, dass das Rauschen zu groß wird, da das Bootstrapping nach jedem Gatter performant durchgeführt werden kann.

Grundsätzlich werden zwei Kryptosysteme unterstützt: Das FHEW und das TFHE Kryptosystem [61]. Die Tabelle 10 zeigt einen Vergleich[1] beider Systeme. Die Spezifikationen der Testumgebung sind in Kapitel 4.1 beschrieben. Auffällig ist, dass TFHE schneller als FHEW ist und weniger Speicherplatz benötigt. Somit wird für die Evaluation das TFHE System bevorzugt verwendet.

| Vergleichsparameter | FHEW | TFHE |
|---|---|---|
| Dauer Bootstrapping NOT | 0,007 ms | 0,007 ms |
| Dauer Bootstrapping AND | 310 ms | 200 ms |
| Dauer Bootstrapping OR | 350 ms | 220 ms |
| Dauer Bootstrapping XOR | 950 ms | 630 ms |
| Dauer Erzeugung Bootstrapping Schlüssel | 22,9 s | 4,4 s |
| Größe Bootstrapping Schlüssel | 1150 MB | 64 MB |

**Tabelle 10:** Vergleich FHWE und TFHE

### 3.1.2 Implementierung

Die Implementierung des Prototyps wurde mit der Krypto-Bibliothek PALISADE [2] durchgeführt. PALISADE ist nicht die einzige Bibliothek, die Funktionen zur gitter-

[1] Es wurde der Durchschnitt aus 100 Durchlaufen ausgewertet und gerundet.
[2] `https://palisade-crypto.org`

basierten Verschlüsselung bereitstellt. Die Tabelle 11 gibt eine Übersicht, über die bekannten Krypto-Bibliotheken, die die gängigen gitterbasierten Verschlüsselungssysteme zur Verfügung stellen. Auffällig ist, dass bis auf Lattigo alle Bibliotheken C++ als Programmiersprache gewählt haben. Alle Bibliotheken (auch Lattigo) benutzen die in C++ geschriebene Bibliothek NTL[3] für performante Operationen mit Vektoren, Polynomen über Ganzzahlen und noch weiteren Körpern und Datenstrukturen.

| **Bibliotheken** | **unterstützte Systeme** | **letzte Version** | **Sprache** |
|---|---|---|---|
| FHEW | FHEW | Mai 17 | C++ |
| HEAAN | CKKS | Sep 18 | C++ |
| HELib | BGV, CKKS | Okt 21 | C++ |
| Lattigo | BFV, CKKS | Jun 22 | Go |
| PALISADE | BGV, BFV, CKKS, FHEW, TFHE | Apr 22 | C++ |
| SEAL | BFV, BGV, CKKS | Mär 22 | C++ |
| TFHE | TFHE | Feb 20 | C++ |

**Tabelle 11:** Übersicht bekannter gitterbasierter Krypto-Bibliotheken

Zudem fällt auf, dass es lediglich drei Bibliotheken gibt, die die gatterbasierten Varianten (FHEW, TFHE) unterstützen. PALISADE ist dabei die einzige Bibliothek, die beides unterstützt. Da die Testläufe zwischen FHEW und TFHE ergeben haben, dass TFHE schneller ist und weniger Speicherplatz benötigt, ist somit lediglich die Bibliothek PALISADE und TFHE in Frage gekommen. Letztendlich hat PALISADE aus drei Gründen überzeugt:

1. PALISADE wird noch aktiv weiterentwickelt und optimiert.
2. PALISADE hat eine deutlich umfangreichere Dokumentation zur Benutzung der Bibliothek.
3. PALISADE unterstützt Gatter im Format *Bristol Fashion*.

Das Format *Bristol Fashion*[4] ermöglicht die Darstellung von Schaltungen in einem Format, welches für Menschen schwierig lesbar ist, allerdings von Maschinen performant eingelesen werden kann. Da der Prototyp der Mehrparteienberechnung ebenfalls mit dem Format umgehen kann, ist es möglich die Evaluierung mit genau dem

[3] `https://libntl.org`
[4] `https://homes.esat.kuleuven.be/~nsmart/MPC/`

gleichen Dokument bzw. der gleichen Schaltung durchzuführen. Auch, wenn die Unterstützung für das Format nur ein optionales Kriterium darstellt, erleichtert es die Evaluierung deutlich, da alle Schaltungen lediglich einmal erstellt bzw. generiert werden müssen.

Der folgende Code-Block zeigt beispielhaft die Funktionsweise des Prototypen. In dem Programm wird die folgende Gleichung voll-homomorph berechnet: 1 AND 1 = 1. Dabei werden alle notwendigen Schritte, auch die Schlüsselerzeugung, Ent- und Verschlüsselung, direkt in dem Programm ausgeführt. Das Bristol Format wird in diesem Beispiel nicht verwendet, da lediglich ein Gatter benutzt wird.

```
#include "binfhecontext.h"
using namespace lbcrypto;

int main() {
  auto cryptoContext = BinFHEContext();
  LWEPlaintext result;

  // Security Level wird auf 128 Bit gesetzt
  // GINX steht für das TFHE System
  cryptoContext.GenerateBinFHEContext(STD128, GINX);

  // Schlüssel werden erzeugt
  auto sk = cryptoContext.KeyGen();

  // Bootstrapping Schlüssel wird erzeugt
  cryptoContext.BTKeyGen(sk);

  // Klartexte (1 und 1) werden verschlüsselt
  auto ciphertextA = cryptoContext.Encrypt(sk, 1);
  auto ciphertextB = cryptoContext.Encrypt(sk, 1);

  // Berechnung wird durchgeführt
  // Bootstrapping wird automatisch durchgeführt.
  auto ciphertextResult =
  cryptoContext.EvalBinGate(AND, ciphertextA, ciphertextB);

  // Ergebnis wird entschlüsselt
```

```
cryptoContext.Decrypt(sk, ciphertextResult, &result);
```

## 3.2 Mehrparteienberechnung

In diesem Abschnitt wird der Prototyp vorgestellt, der auf der Mehrparteienberechnung basiert.

### 3.2.1 Architektur

Die Architektur des Prototypen für die Mehrparteienberechnung basiert auf dem Twin-Cloud-Modell [58]. Im Folgenden wird das Modell kurz vorgestellt. Grundsätzlich nutzt das Twin-Cloud-Modell eine hybride Cloud mit dem Garbled Circuits Protokoll. Das Modell besteht aus zwei Phasen. Zunächst erfolgt die Setup-Phase, die in Abbildung 16 zu sehen ist.

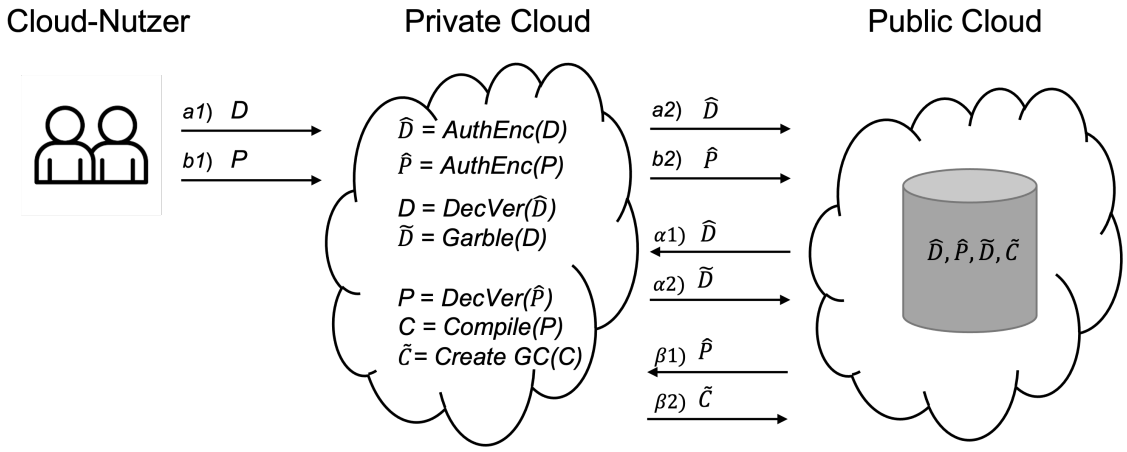



**Abbildung 16:** Setup-Phase des Twin-Cloud-Modell (Quelle: Eigene Zeichnung angelehnt an [58])

Die Phase wird genutzt, um alle notwendigen Schritte für die eigentliche Nutzung, der Query-Phase, durchzuführen. Dazu gehören die folgenden vier Schritte, wobei alle Aktionen des Cloud-Nutzers mit lateinischen Buchstaben dargestellt werden und automatisch die Aktionen auslösen, die in griechischen Buchstaben dargestellt sind.

**Änderung der Daten:** Wenn der Cloud-Nutzer neue oder abgeänderte Daten $D$ in der Cloud berechnen möchte (*Schritt a1*), dann wird $D$ als $\hat{D} =$

AuthEnc($D$) verschlüsselt in der Public Cloud gespeichert (*Schritt a2*). Wenn die Daten $D$ geändert werden, dann werden die verschleierten Daten $\widetilde{D}$ automatisch erneut verschleiert und alle dazugehörigen GC $\widetilde{C}$ gelöscht.

**Änderung des Programms:** Wenn der Cloud-Nutzer eine neues oder abgeändertes Programm $P$ in der Cloud berechnen möchte (*Schritt b1*), dann wird $P$ als $\hat{P} = \text{AuthEnc}(P)$ verschlüsselt in der Public Cloud gespeichert (*Schritt b2*). Wenn das Programm $P$ geändert wird, dann werden alle dazugehörigen GC $\widetilde{C}$ gelöscht.

**Verschleiern der Daten:** Wenn die Daten $D$ geändert werden, müssen die verschleierten Daten $\widetilde{D}$ erneut generiert werden. Zu diesem Zweck fordert die Private Cloud die sicher gespeicherten Daten $\hat{D}$ von der Public Cloud (*Schritt* $\alpha 1$) an, stellt die Daten $D = \text{DecVer}(\hat{D})$ wieder her, generiert die entsprechenden verschleierten Daten $\widetilde{D} = \text{Garble}(D)$ und speichert diese zurück in die Public Cloud (*Schritt* $\alpha 2$).

**Verschleiern des Programms:** Immer wenn die Daten $D$ oder das Programm $P$ geändert werden oder die Private Cloud Kapazitäten für Vorberechnungen hat, werden neue GC erzeugt. Dazu fordert die Private Cloud das sicher gespeicherte Programm $\hat{P}$ von der Public Cloud an (*Schritt* $\beta 1$), stellt das Programm $P = \text{DecVer}(\hat{P})$ wieder her, kompiliert es in eine boolesche Schaltung $C = \text{Compile}(P)$, erzeugt einen neuen GC $\widetilde{C} = \text{Garble}(C)$ und speichert diesen zurück in die Public Cloud (*Schritt* $\beta 2$).

In der Query-Phase schickt der Cloud-Nutzer Anfragen $q$ an die Private Cloud, die mit $r = P(q, D)$ beantwortet werden. Die Abbildung 17 zeigt den Ablauf der Query-Phase.

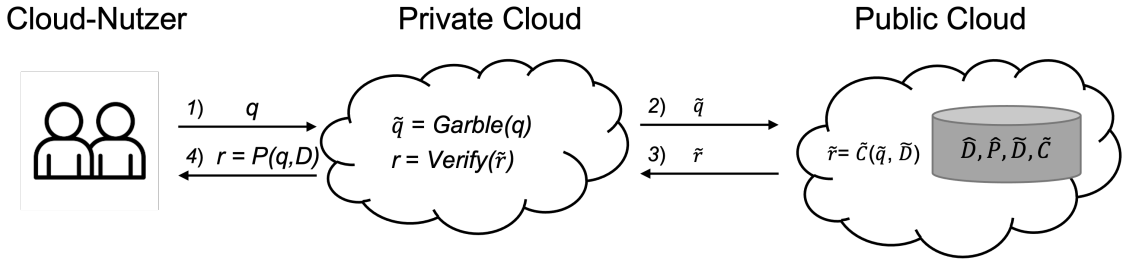



**Abbildung 17:** Query-Phase des Twin-Cloud-Modell (Quelle: Eigene Zeichnung angelehnt an [58])

Insgesamt werden pro Anfrage vier Schritte ausgeführt:

1. Der Cloud-Nutzer schickt eine Anfrage $q$ an die Private Cloud.
2. Die Private Cloud verschleiert die Anfrage mit $\widetilde{q}$ =Garble($q$) und sendet $\widetilde{q}$ an die Public Cloud.
3. Die Public Cloud berechnet die verschleierte Antwort $\widetilde{r} = \widetilde{C}(\widetilde{q}, \widetilde{D})$ und sendet $\widetilde{r}$ an die Public Cloud.
4. Die Private Cloud verifiziert die Antwort mit $r$ = Verify($\widetilde{r}$) und sendet $r = P(q, D)$ an den Cloud-Nutzer.

Der entwickelte Prototyp für die Mehrparteienberechnung nutzt das Twin-Cloud-Modell lediglich als Vorlage. Das Modell wurde wie im folgenden Abschnitt beschrieben für die Entwicklung des Prototyps angepasst. Das Twin-Cloud-Modell wurde von den Autoren als Framework entwickelt, um jegliches Programm auszuführen. Das Modell wandelt mit Hilfe des Circuit Compiler jeglichen Programmcode von einer klassischen Programmiersprache (z.B. C oder Java) in eine Schaltung um. Diese Funktionalität wird für die Beantwortung der Forschungsfrage nicht benötigt und wurde daher nicht mit umgesetzt. Dementsprechend fallen alle $b$ und $\beta$ Schritte, ausgenommen Teile von $\beta 2$ weg. Zudem wird die Speicherungsfunktionalität in der Public Cloud nicht benötigt, da für die Beantwortung der Forschungsfrage nur ausgewählte Szenarien untersucht werden, die ohne Probleme in der Private Cloud gespeichert werden können. Die letzte Änderung ist der Verzicht auf die Abfragen des Cloud-Nutzers. Der Prototyp ist so entwickelt, dass die Private Cloud die Funktionen des Cloud-Nutzers übernimmt. Grund dafür ist die Verringerung der Komplexität, da es für die Beantwortung der Forschungsfrage lediglich einen Cloud-Nutzer bedarf. Die Abbildung 18 zeigt das modifizierte Twin-Cloud-Modell, welches als Grundlage für die Architektur des Prototypen herangezogen wurde.

Zur besseren Übersicht wurden beide Phasen in einer Abbildung dargestellt. Auffällig ist, dass die Setup-Phase durch die oben beschriebenen Anpassungen deutlich schlanker ausfällt. An der Query-Phase hat sich im Grunde, bis auf die Integration des Cloud-Nutzers in die Private Cloud, nichts geändert.

Der größte Unterschied zwischen dem klassischen Garbled Circuits Protokoll und dem angepassten Twin-Cloud-Modell ist, dass beim klassischen GC Protokoll beide Seiten gleichberechtigt sind. Das bedeutet, dass sowohl beide Seiten eine Eingabe haben als auch das Ergebnis der Berechnung erfahren. Dies ist im angepassten Twin-Cloud-Modell nicht vorgesehen, da die Vertraulichkeit der Daten gewährleistet werden soll. Die Private Cloud wird bevorzugt behandelt, sodass nur diese das Ergebnis der Berechnung erfährt.

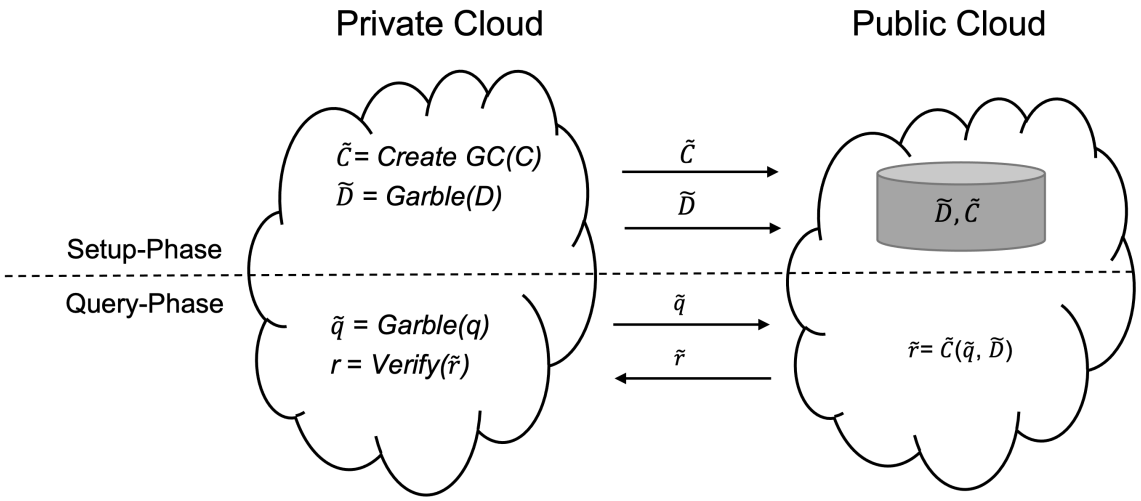



**Abbildung 18:** Beide Phasen des angepasstesten Twin-Cloud-Modells (Quelle: Eigene Zeichnung)

### 3.2.2 Implementierung

Der Prototyp nutzt als Grundlage das EMP-Tool [62] als Grundlage. Dieses stellt alle notwendigen Funktionen zur Nutzung von Garbled Circuits bereit. Dazu zählen der Verbindungsaufbau zwischen zwei Parteien, das Abarbeiten des Protokolls aus Sicht des Generators und Evaluators und der Oblivious Transfer. Der Oblivious Transfer nutzt keine Post-Quantum-Kryptographie, sodass dieser gegen einen quantenresistenten OT ausgetauscht werden muss. Dafür wurde der Post-Quantum-Oblivious-Transfer von [47] benutzt. Dieser ist in Abschnitt 2.3.4 beschrieben.

Die folgende beiden Code-Blöcke zeigen Ausschnitte, um die Funktionsweise des Prototypen zu verstehen. Als Beispiel wird die Addition zweier Zahlen (500 und 1000) über eine zuvor im Bristol Format definierte Schaltung gezeigt. Dabei müssen beide Programme parallel ausgeführt werden, damit die Verbindung aufgebaut werden kann. Die Alice Partei bildet die Private Cloud ab, welche die Funktion des Generators darstellt. Die Bob Partei bildet die Public Cloud ab, welche die Funktion des Evaluators darstellt. Das Beispielprogramm von Alice sieht wie folgt aus:

```
// Alice (private Cloud) öffnet Verbindung über Port 444 zu Bob
io = new NetIO(party==ALICE?nullptr:"192.168.1.15", 444);

// Schaltung im Bristol Format wird geladen und verschleiert
cf = new CircuitFile("Speicherort");
```

```
// Einlesen der Zahlen, die addiert werden sollen
Integer a(500, ALICE);
Integer b(1000, BOB);

// Output wird vorbereitet
Integer c(0, OUTPUT);

// Quantenresistenter OT wird vorbereitet
ot = new PQOT(io, ALICE);
ot->keygen();
ProtocolExecution::prot_exec->do_batched_ot();

// Auf die Verbindung mit Bob (Public Cloud) wird gewartet
io->sync();

// Verschleierte Schaltung wird an Bob verschickt
ProtocolExecution::prot_exec = new Generator(io, cf)

// Schaltung wird Gatter für Gatter abgearbeitet
cf->compute(c.bits, a.bits, b.bits);

// Alice erhält Ergebnis
auto output = c.reveal(ALICE);

//Verbindung wird geschlossen
delete io;
```

Das Beispielprogramm von Bob sieht wie folgt aus:

```
// Bob (Public Cloud) öffnet Verbindung über Port 444 für Alice
io = new NetIO(party==BOB?nullptr:"192.168.1.15", 444);

// Quantenresistenter OT wird vorbereitet
ot = new PQOT(io, BOB);
ProtocolExecution::prot_exec->do_batched_ot();

// Auf die Verbindung mit Alice (Private Cloud) wird gewartet
io->sync();
```

```
// Verschleierte Schaltung wird empfangen
ProtocolExecution::prot_exec = new Evaluator(io, cf)

// Schaltung wird Gatter für Gatter abgearbeitet
cf->compute();


//Verbindung wird geschlossen
delete io;
```

### 3.3 Grenzen und Ausblick beider Prototypen

Beide Prototypen berücksichtigen nicht, dass die Schlüsselerzeugung, Ver- und Entschlüsselung und Verschleierung der Abfrage eigentlich beim Cloud-Nutzer durchgeführt werden müsste, um die Vertraulichkeit zu wahren. Grund für die Vernachlässigung dieser Gegebenheit ist, dass die Auslagerung zusätzliche Komplexität mit sich bringt. Für die Evaluation beider Verfahren ist die Implementierung einer zusätzlichen Schnittstelle kaum ausschlaggebend, da bei beiden Verfahren diese Schnittstelle im Prototypen nicht vorhanden ist. Der Fokus soll auf den Verfahren an sich und nicht auf der Übertragung der Daten liegen.

Nichtsdestotrotz ist diese Ausbaustufe interessant, da die Verfahren nicht die gleichen Daten schicken und es somit zu unterschiedlichen Geschwindigkeiten bei der Übertragung kommen kann. Des Weiteren ist die Schlüsselerzeugung eine rechenintensive Operation. Es ist also durchaus interessant die Performance mit verschiedenen Endgeräten, wie z.B. einem Smartphone, einem IoT-Gerät oder einem Raspberry Pi, beim Cloud-Nutzer zu untersuchen. Es muss auch bedacht werden, dass die Anforderungen an die Tunnelverbindung dieselben sind, wie die für die bisherigen Prototypen. Die Tunnelverbindung muss die Vertraulichkeit und Quantenresistenz gewährleisten und somit bedarf es unteranderem einen quantenresistenten Schlüsselaustausch, um eine symmetrisch verschlüsselte Verbindung herzustellen.

Bei dem Prototypen, der die Mehrparteienberechnung abbildet, gibt es noch Optimierungspotential hinsichtlich des Garbled Circuit Protokolls. Es wurde lediglich die Optimierung Point & Permut [48] verwendet, da diese bereits in den verwendeten Bibliotheken unterstützt wurde. In Abschnitt 2.3.5 sind weitere mögliche Optimierungen aufgelistet, die die Anzahl der zu versendeten Nachrichten deutlich senken

können. Dies würde sich positiv auf die Geschwindigkeit und die Größe des Netzwerkverkehrs auswirken.

Die für den Vergleich herangezogenen Schaltungen wurde nicht spezifisch auf eine Technologie optimiert, da eine optionale Voraussetzung ist, dass die Schaltungen identisch sind. Für den Prototypen, der die gatterbasierte homomorphe Verschlüsselung abbildet, hat die Art des Gatters einen deutlichen Einfluss auf die Dauer des Bootstrapping wie in Tabelle 10 erkennbar. So benötigt ein NOT-Gatter durchschnittlich lediglich $0,007$ ms, während ein AND-Gatter oder OR-Gatter im Durchschnitt $200 - 220$ ms benötigen. Das Gatter mit der längsten Durchlaufzeit, XOR-Gatter mit durchschnittlich 630 ms, ist genau das Gatter, welches beim Prototypen der Mehrparteienberechnung durch die point & permute Technik [48] nur ein Viertel der Nachrichten zum Abarbeiten benötigt. Das bedeutet, dass die gleiche Schaltung (im gleichen Format) lediglich für eine der beiden Technologien optimiert werden kann. Für den Vergleich wurde daher keine Optimierung durchgeführt. Es wäre trotzdem spannend den Vergleich mit zwar den gleichen Schaltungen, allerdings in zwei jeweils für die Technologien optimierten Formaten, durchzuführen.

## 3.4 Bewertungskriterien

In diesem Abschnitt werden die Bewertungskriterien definiert, die für die Evaluation und schließlich der Beantwortung der Forschungsfrage entscheidend sind. Dazu werden zunächst die Voraussetzungen definiert.

Bevor die Evaluation stattfinden kann, bedarf es verschiedener Voraussetzungen, die bei Erfüllung einen gewissen Qualitätsstandard garantieren. Es wird in zwei Kategorien von Voraussetzungen unterschieden. Die obligatorischen und die optionalen Voraussetzungen. Die obligatorischen Voraussetzungen stellen sicher, dass der Vergleich beider Technologien zur Forschungsfrage passt. Die optionalen Voraussetzungen erhöhen die Aussagekraft des Vergleichs.

Die folgenden Voraussetzungen sind **obligatorisch**:

1. **Quantenresistent:** Der verwendete Prototyp muss auf einer Post-Quantum Technologie beruhen.
2. **Vertraulich:** Die Vertraulichkeit der Daten muss sichergestellt sein.
3. **mehrfache Ausführung:** Die Anzahl der Testläufe muss statistisch signifikant sein. Statistisch signifikant meint, dass bei einer Effektstärke von 0,2

und einer statistischen Power von 0,8 das Niveau von $\alpha$ mindestens bei 5% in einem einseitigen ungepaarten t-Test liegen muss[5].

4. **identische Cloud-Parameter:** Die auswählbaren Parameter in der Cloud, die unabhängig der benötigten Ressourcen für den Betrieb der Prototypen sind, müssen bei der Kalkulation der Kosten identisch gewählt oder gar nicht berücksichtigt werden. Zu jenen Parametern zählen beispielsweise die Reservierungskosten für die Bereitstellung, der Standort des Rechenzentrums oder das Bezahlmodell (z.B. per Vorkasse).

Die folgenden Voraussetzungen sind **optional**:

1. **Betrieb in der Cloud:** Der Prototyp wird nicht lokal, sondern in einer Cloud-Umgebung betrieben.
2. **Identische Schaltungen:** Die Schaltungen, die beim Vergleich beider Technologien verglichen werden, sind identisch.
3. **verschiedene Schaltungen:** Es wird nicht nur ein Vergleich ausgeführt, sondern mehrere Vergleiche werden über verschiedene Schaltungen durchgeführt.
4. **verschiedene Cloud-Anbieter:** Die Kostenkalkulation wird auf Basis verschiedener Cloud-Anbieter durchgeführt.

Für die Evaluation ist auf Grund der Forschungsfrage der Preis für die Ausführung beider Technologien interessant. Daher müssen beim Vergleich folgende Parameter betrachtet werden:

1. **CPU:** Hierzu zählen die Kosten für die Bereitstellung der CPU (inkl. RAM). Durch das On-Demand-Modell lässt sich ein Preis pro Minute für die Bereitstellung ermitteln.
2. **Speicher:** Hierzu zählen die Kosten für die Bereitstellung der Festplatte (persistenter Speicher) und dem Arbeitsspeicher (flüchtiger Speicher).
3. **Netzwerkverkehr:** Hierzu zählt der ein- und ausgehende Netzwerkverkehr. Durch das On-Demand-Modell lässt sich ein Preis pro Gigabyte Netzwerkverkehr ermitteln.

[5]Die Werte stammen aus den Empfehlungen von [63]

Aus dem Vergleich beider Verfahren in Kapitel 2.5.3 ist hervorgegangen, dass die homomorphe Verschlüsselung vor allem auf der Rechenleistung basiert und die Mehrparteienberechnung auf dem Versand von Nachrichten, also netzwerkbasiert arbeitet. Aus den aufgezählten Kostenparametern kann also abgeleitet werden, dass das Augenmerk auf dem Vergleich zwischen den CPU-Kosten und Netzwerkverkehr-Kosten liegt, da dies die jeweils größten Kostenpositionen der Verfahren darstellen.

# 4 Evaluation der Verfahren

In diesem Kapitel werden die beiden Verfahren gemäß der Forschungsfrage miteinander verglichen. Dazu werden zunächst die Rahmenbedingungen erläutert. Anschließend werden die Szenarien und deren Implementierungen vorgestellt. Das Kapitel schließt mit der Durchführung und der Bewertung der Evaluation ab.

## 4.1 Rahmenbedingungen

In diesem Abschnitt werden die Rahmenbedingungen für die Evaluation beider Prototypen vorgestellt.

### 4.1.1 Homomorphe Verschlüsselung

Der Prototyp der homomorphen Verschlüsselung wird in der Amazon Cloud in dem Dienstmodell IaaS betrieben. Dazu wird eine Instanz des Typs *t3a.xlarge* im Service Amazon Elastic Compute Cloud (Amazon EC2) benutzt. Die Tabelle 12 listet die Spezifikationen der Instanz auf. Die Wahl der Instanz fiel auf Grund der ähnlichen Spezifikationen und Benchmark-Ergebnissen[1] zum *Thinkpad T470s*, auf welchem der Prototyp der Mehrparteienberechnung durchgeführt wird, auf den Typ *t3a.xlarge*. Durch den Betrieb in der Cloud wird die erste optionale Voraussetzung erfüllt.

Der homomorphe Prototyp wurde mit einer 128-Bit Verschlüsselung in allen Szenarien durchgeführt.

[1] sysbench ist ein Programm zum Erstellen eines CPU-Benchmark; verfügbar unter `https://openbenchmarking.org/test/pts/sysbench`

| Spezifikationen | Ausprägungen |
|---|---|
| Name | t3a.xlarge |
| vCPUs | 4 |
| RAM | 16 GB |
| Speicher | 30 GB (SSD gp2) |
| sysbenchmark | 1206 events/s |

**Tabelle 12:** Spezifikationen t3a.xlarge

### 4.1.2 Mehrparteienberechnung

Der Prototyp der Mehrparteienberechnung wird nicht in der Cloud, sondern auf zwei *Thinkpad T470s* betrieben. Die Tabelle 13 listet die Spezifikationen der beiden Laptops auf. Durch die lokale Ausführung wird die erste optionale Voraussetzung nicht erfüllt.

| Spezifikationen | Ausprägungen |
|---|---|
| Name | Thinkpad T470s |
| CPUs | 4 |
| RAM | 16 GB |
| Speicher | 256 GB (SSD) |
| sysbenchmark | 1192 events/s |

**Tabelle 13:** Spezifikationen Thinkpad T470s

Hauptausschlaggebend für die lokale Ausführung ist die Manipulation des Netzwerks und geringere Komplexität. Das lokale Netzwerk wird auf 1 GBit/s begrenzt, da gerade in Deutschland eine höhere Bandbreite oftmals nicht vorhanden ist. Durch die lokale Ausführung bedarf es keiner weiteren Sicherheitsmaßnahmen, wenn beispielsweise der Port für die Verbindung zwischen Alice und Bob geöffnet wird. Bei der Ausführung in der Cloud bedarf es weiterer Schutzmechanismen, da die verwendeten IP-Adressen öffentlich sind.

## 4.2 Szenarien

Um die Evaluation durchzuführen, werden drei Szenarien definiert, um die dritte optionale Voraussetzung (verschiedene Schaltungen) zu erfüllen. Die Szenarien unterscheiden sich lediglich in dem verwendeten Schaltkreis. Die Szenarien sind wie folgt definiert:

1. **AES:** Ein beliebiger Input soll mit AES 128-Bit verschlüsselt werden.
2. **Addition:** Zwei 32 Bit-Zahlen sollen miteinander addiert werden.
3. **Multiplikation:** Zwei 32 Bit-Zahlen sollen miteinander multipliziert werden.

Die Tabelle 14 zeigt jeweils die Größe der Ein- und Ausgaben, sowie die Anzahl der jeweiligen Gatter.

| **Szenario** | **Eingabe** | **Ausgabe** | **# AND** | **# XOR** | **# NOT** |
|---|---|---|---|---|---|
| AES | 128 Bit | 128 Bit | 6800 | 25124 | 1692 |
| Addition | 32 Bit | 33 Bit | 127 | 61 | 187 |
| Multiplikation | 32 Bit | 64 Bit | 265 | 115 | 379 |

**Tabelle 14:** Vergleich der Szenarien

## 4.3 Durchführung

Der folgende Abschnitt stellt die Ergebnisse der Durchführung dar. Alle Szenarien wurden 310-mal pro Prototyp ausgeführt, damit die obligatorische Voraussetzung der statischen Signifikanz erfüllt ist, da $\alpha = 0,05$ bei einer Stichprobengröße von 310 ist, wenn die Effektstärke 0,2 und die statische Power 0,8 beträgt.

### 4.3.1 Homomorphe Verschlüsselung

Der folgende Abschnitt stellt die Ergebnisse des homomorphen Prototyps für die drei ausgewählten Szenarien dar. Alle Durchlaufzeiten sind in Millisekunden angegeben. Die Tabelle 15 zeigt die Zusammenfassung der untersuchten Parameter für die jeweiligen Szenarien. Im Anhang A.1 sind die jeweiligen Messergebnisse für interessierte

Leser detailliert (inklusive Festplattenspeicher, flüchtigem Speicher, Standardabweichung, Varianz) aufgeführt.

| **untersuchte Parameter** | **AES** | **Addition** | **Multiplikation** |
|---|---|---|---|
| Durchlaufzeit | 9.548.705,00 ms | 72.290,87 ms | 1.343.027,20 ms |
| Netzwerkverkehr (ausgehend) | - | - | - |

**Tabelle 15:** Zusammenfassung der Messergebnisse für die jeweiligen Szenarien für den Prototypen homomorphe Verschlüsselung

### 4.3.2 Mehrparteienberechnung

Der folgende Abschnitt stellt die Ergebnisse des Mehrparteienberechnung-Prototyps für die drei ausgewählten Szenarien dar. Alle Durchlaufzeiten sind in Millisekunden angegeben. Der ausgehende Netzwerkverkehr wird in Kilobyte angegeben.

Die Tabelle 16 zeigt die Zusammenfassung der untersuchten Parameter für die jeweiligen Szenarien. Im Anhang A.2 sind die jeweiligen Messergebnisse für interessierte Leser detailliert (inklusive Festplattenspeicher, flüchtigem Speicher, Standardabweichung, Varianz) aufgeführt.

| **untersuchte Parameter** | **AES** | **Addition** | **Multiplikation** |
|---|---|---|---|
| Durchlaufzeit | 359,63 ms | 60,42 ms | 110,38 ms |
| Netzwerkverkehr (ausgehend) | 5.318 KB | 769 KB | 1.828 KB |

**Tabelle 16:** Zusammenfassung der Messergebnisse für die jeweiligen Szenarien für den Prototypen homomorphe Verschlüsselung

## 4.4 Bewertung

Der folgende Abschnitt bewertet die beiden Prototypen hinsichtlich der Kostenstruktur. Dementsprechend wird in diesem Abschnitt die Forschungsfrage beantwortet. Bevor die Bewertung durchgeführt wird, werden zunächst die in Kapitel 3.4 definierten Voraussetzungen überprüft. Die Tabelle 17 listet den Erfüllungsgrad der jeweiligen Voraussetzung. Alle obligatorischen Voraussetzungen sind erfüllt. Bei den optionalen Voraussetzungen sind bis auf eine Voraussetzung alle voll erfüllt.

| **Voraussetzung** | **Relevanz** | **Erfüllungsgrad** | **Begründung** |
|---|---|---|---|
| Quantenresistent | obligatorisch | voll erfüllt | Die Quantenresistenz beider Prototypen ist auf Grund der verwendeten Verfahren sichergestellt. |
| Vertraulich | obligatorisch | voll erfüllt | Die Vertraulichkeit der Daten ist durch die Architektur beider Prototypen sichergestellt. |
| mehrfache Ausführung | obligatorisch | voll erfüllt | Beide Prototypen wurden 310-mal ausgeführt. |
| identische Cloud-Parameter | obligatorisch | voll erfüllt | Es wurden identische Cloud-Parameter bei der Kostenkalkulation gewählt. |
| Betrieb in der Cloud | optional | teilweise erfüllt | Der homomorphe Prototyp wurde in der Cloud betrieben. Der Prototyp der Mehrparteienberechnung nicht. |
| identische Schaltungen | optional | voll erfüllt | Es wurden bei der Evaluation identische Schaltungen verwendet. |
| verschiedene Schaltungen | optional | voll erfüllt | Es wurden drei verschiedene Schaltungen evaluiert. |
| verschiedene Cloud-Anbieter | optional | voll erfüllt | Es wurden drei Cloud-Anbieter bei der Kostenkalkulation berücksichtigt. |

**Tabelle 17:** Überprüfung der Voraussetzungen

### 4.4.1 Kostenvergleich für verschiedene Cloud-Provider

Die Gesamtkosten für den Betrieb in der Cloud lassen sich über die folgende Gleichung ermitteln:

$$\text{Gesamtkosten} = \text{CPU-Kosten} + \text{Speicher-Kosten} + \text{Netzwerkkosten} \tag{4.1}$$

**Formel 4.1:** Gesamtkosten des Cloud-Betriebs

Die konkreten Kosten hängen vom jeweiligen Cloud-Anbieter ab. Im Folgenden werden die Gesamtkosten am Beispiel der Cloud-Anbieter Amazon, Google und Microsoft aufgeführt. Die Kosten werden jeweils für 1000 Durchgänge[2] mit der durchschnittlichen Laufzeit berechnet. Für den Prototypen der Mehrparteienberechnung werden lediglich die Kosten für die Public Cloud berücksichtigt. Am Beispiel der Amazon Cloud wird die Berechnung der Kosten ausführlich vorgestellt. Die Berechnung der Kosten für die Microsoft und Google Cloud ist zusammengefasst.

#### 4.4.1.1 Amazon Cloud

Die Kostenparameter sind in der Amazon Cloud wie folgt ausgeprägt:

**CPU:** \$0,1504 pro Stunde

**Speicher:** Persistenter Speicher (30 GB) \$3 pro Monat; flüchtiger Speicher in den CPU-Kosten enthalten

**Netzwerkverkehr:** \$0,09 pro ausgehenden GB

Nun werden die Gesamtkosten beider Prototypen in den drei Szenarien gegenübergestellt.

**AES** Die Gesamtkosten für den homomorphen Prototypen betragen \$409,80, während die Gesamtkosten für den Prototyp der Mehrparteienberechnung lediglich \$0,49 betragen. Die Tabelle 18 listet die einzelnen Kosten der beiden Verfahren auf.

[2] Zu besseren Lesbarkeit

| Parameter | homomorph | Mehrparteienberechnung |
|---|---|---|
| CPU | $398,92 | $0,01 |
| Speicher | $10,87 | $0,00 |
| Netzwerk | $0,00 | $0,48 |
| Gesamtkosten | $409,80 | $0,49 |

**Tabelle 18:** Kosten für den Betrieb für das Szenario AES mit dem homomorphen Prototypen in der Amazon Cloud

**Addition** Die Gesamtkosten für den homomorphen Prototypen betragen $3,10, während die Gesamtkosten für den Prototyp der Mehrparteienberechnung lediglich $0,07 betragen. Die Tabelle 19 listet die einzelnen Kosten auf.

| Parameter | homomorph | Mehrparteienberechnung |
|---|---|---|
| CPU | $3,02 | $0,00 |
| Speicher | $0,08 | $0,00 |
| Netzwerk | $0,00 | $0,07 |
| Gesamtkosten | $3,10 | $0,07 |

**Tabelle 19:** Kosten für den Betrieb für das Szenario Addition mit dem homomorphen Prototypen in der Amazon Cloud

**Multiplikation** Die Gesamtkosten für den homomorphen Prototypen betragen $57,64, während die Gesamtkosten für den Prototyp der Mehrparteienberechnung lediglich $0,17 betragen. Die Tabelle 20 listet die einzelnen Kostenparameter auf.

| Parameter | homomorph | Mehrparteienberechnung |
|---|---|---|
| CPU | \$56,11 | \$0,01 |
| Speicher | \$1,53 | \$0,00 |
| Netzwerk | \$0,00 | \$0,16 |
| Gesamtkosten | \$57,64 | \$0,17 |

**Tabelle 20:** Kosten für den Betrieb für das Szenario Multiplikation mit dem homomorphen Prototypen in der Amazon Cloud

#### 4.4.1.2 Google Cloud

Die Kostenparameter sind in der Google Cloud wie folgt ausgeprägt:

**CPU:** \$0,1304 pro Stunde

**Speicher:** Persistenter Speicher (30 GB) \$3 pro Monat; flüchtiger Speicher in den CPU-Kosten enthalten

**Netzwerkverkehr:** \$0,12[3] pro ausgehenden GB

Nun werden die Gesamtkosten beider Prototypen in den drei Szenarien gegenübergestellt, wobei die einzelnen Kostenparamter nicht wie bei der Amazon Cloud explizit aufgeführt werden. Die Tabelle 21 listet die Gesamtkosten der verschiedenen Szenarien auf.

| Szenario | homomorph | Mehrparteien |
|---|---|---|
| AES | \$356,75 | \$0,65 |
| Addition | \$2,70 | \$0,09 |
| Multiplikation | \$50,18 | \$0,22 |

**Tabelle 21:** Gesamtkostenüberstellung der beiden Prototypen in der Google Cloud

[3] Variiert von Region zu Region. Der angegebene Preis ist für Traffic an Ziele in der ganzen Welt (ausgenommen China und Australien)

#### 4.4.1.3 Microsoft Cloud

Die Kostenparameter sind in der Microsoft Cloud wie folgt ausgeprägt:

**CPU:** $0,234 pro Stunde

**Speicher:** Persistenter Speicher (32 GB) und flüchtiger Speicher in den CPU-Kosten enthalten

**Netzwerkverkehr:** $0,05 pro ausgehenden GB

Nun werden die Gesamtkosten beider Prototypen in den drei Szenarien gegenübergestellt, wobei die einzelnen Kostenparamter nicht wie bei der Amazon Cloud explizit aufgeführt werden. Die Tabelle 22 listet die Gesamtkosten der verschiedenen Szenarien auf.

| **Szenario** | **homomorph** | **Mehrparteien** |
|---|---|---|
| AES | $620,67 | $0,29 |
| Addition | $4,70 | $0,04 |
| Multiplikation | $87,30 | $0,10 |

**Tabelle 22:** Gesamtkostenüberstellung der beiden Prototypen in der Microsoft Cloud

### 4.4.2 Fazit

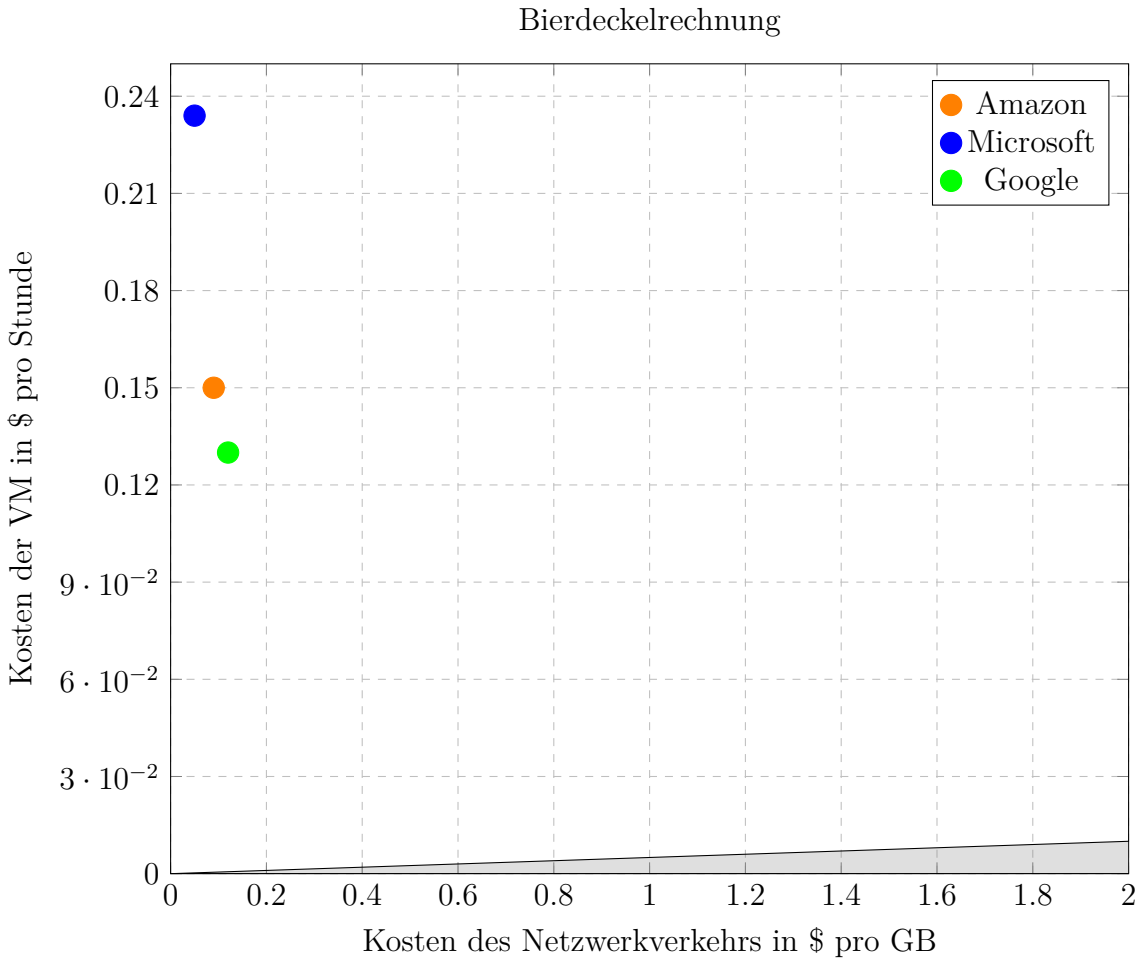
Der Kostenvergleich aller Szenarien und aller Cloud-Provider entscheidet sich immer zugunsten des Prototypen der Mehrparteienberechnung. Somit ist die Forschungsfrage klar beantwortet: Die Mehrparteienberechnung ist preiswerter. Wie viel preiswerter die Mehrparteienberechnung ist, hängt vom Cloud Provider und Szenario ab. In den durchgeführten Vergleichen schwankte der Faktor von 28 (Szenario Addition in der Google Cloud) bis 2145 (Szenario AES in der Microsoft Cloud). Dies bedeutet, dass für den Preis einer AES-Verschlüsselung in der Microsoft-Cloud mit den homomorphen Verfahren, 2145 AES-Verschlüsselungen durchgeführt werden könnten, wenn stattdessen die Mehrparteienberechnung benutzt wird.

Dieses Ergebnis deckt sich mit den Vergleichen, wenn die Anforderung der Quantenresistenz nicht berücksichtigt wird. Auch in diesem Fall ist die Mehrparteienberechnung deutlich effizienter [58].

Welches Verfahren grundsätzlich günstiger ist, hängt eng mit der Preispolitik der Cloud-Provider zusammen. Auf Grund des aktuellen Verhältnisses von CPU-Kosten und Netzwerkverkehr-Kosten ist die Mehrparteienberechnung immer günstiger. Die folgende „Bierdeckelrechnung“ stellt ein Verhältnis bereit, welche eine erste Einschätzung ermöglicht, um zu ermitteln, welches Verfahren preisweiter in Bezug auf die Kostenparameter ist. Dabei werden bewusst bestimmte Kostenparameter vernachlässigt, um die Komplexität gering zu halten. Als Basis dient das Szenario Multiplikation. Bei der Bierdeckelrechung werden die CPU- und Speicherkosten ($K_R$) mit den Kosten des Netzwerkverkehrs ($K_N$) ins Verhältnis gesetzt.

$$\frac{K_R}{K_N * 0{,}005} = \begin{cases} > 1 & \text{Mehrparteienberechnung ist günstiger} \\ = 1 & \text{Kosten identisch} \\ < 1 & \text{Homomorphes Verfahren ist günstiger} \end{cases}$$

Die folgende Grafik zeigt das Verhältnis der beiden Kostenparameter. Befindet sich der ausgewählte Cloud-Provider oberhalb der schwarzen Linie bzw. im weißen Bereich, dann ist die Mehrparteienberechnung günstiger. Ist der ausgewählte Cloud-Provider unter der schwarzen Linie bzw. im grau eingefärbten Bereich, dann ist das homomorphe Verfahren günstiger.


Bierdeckelrechnung
Amazon
Microsoft
Google
Kosten der VM in $ pro Stunde
0.24
0.21
0.18
0.15
0.12
$9 \cdot 10^{-2}$
$6 \cdot 10^{-2}$
$3 \cdot 10^{-2}$
0
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
Kosten des Netzwerkverkehrs in $ pro GB

# 5 Zusammenfassung

Diese Thesis wird eingeleitet mit der Problemstellung rundum die Cloud-Nutzung. Konkret ist die Vertraulichkeit der in der Cloud verarbeitenden Daten nicht gewährleistet. Möchte man die Cloud nutzen, allerdings nicht auf die Vertraulichkeit verzichten, so gibt es verschiedene Ansätze, die dies ermöglichen. Damit auch in der Zukunft, wenn das erste asymmetrische Kryptosystem praktisch durch Quantencomputer gebrochen wurde, die Lösung noch relevant ist, wird eine weitere Anforderung an die Verfahren gestellt: Sie sollen zusätzlich quantenresistent sein. Somit bleiben lediglich zwei Verfahren übrig, welche die quantenresistente vertrauliche Cloud-Nutzung ermöglichen: Die homomorphe Verschlüsselung auf Basis eines gitterbasierten Kryptosystems und die sichere Mehrparteienberechnung auf Basis der Garbled Circuits. Ziel der Thesis war es zu ermitteln, welches dieser beiden Verfahren in der Praxis preiswerter zu betreiben ist. Um diese Evaluation durchführen zu können, wurde jeweils ein Prototyp pro Verfahren entwickelt.

Dazu wurden zunächst die Grundlagen geschaffen, um die beiden verwendeten Verfahren erläutern zu können. Anschließend wurden beide Verfahren vorgestellt und theoretisch verglichen. Dabei kam heraus, dass die gitterbasierte Verschlüsselung rechenbasiert und die Mehrparteienberechnung netzwerkbasiert arbeitet. Entsprechend hoch ist die Praxisrelevanz, da in der Cloud typischerweise nicht nur die CPU-Minuten berechnet werden, sondern auch der ausgehende Netzwerkverkehr. Das Augenmerk liegt auf dem Vergleich beider Kostenparameter.

Anschließend wurde die Architektur beider Verfahren bzw. beider Prototypen vorgestellt. Dabei nutzt der Prototyp der homomorphen Verschlüsselung die Krypto-Bibliothek PALISADE als Grundlage. Für die Evaluierung wurde das Kryptosystem TFHE implementiert. Der Prototyp der Mehrparteienberechnung nutzt als Architekturvorlage das Twin Clouds Modell. Implementiert wurde dies mit dem EMP-Tool, wobei der Oblivious Transfer durch einen quantenresistenten OT ersetzt wurde, damit die Anforderung an die Quantenresistenz erfüllt ist. Beide Prototypen stellen lediglich die Basisfunktionalitäten dar und verzichten auf verschiedene Funktionen, die für einen praxisrelevanten Betrieb notwendig sind. Zudem gibt es bei beiden

Prototypen Optimierungspotential. Die entsprechende selbstkritische Beleuchtung und Architekturentscheidung wurde ausführlich beschrieben.

Um den Vergleich durchführen zu können, wurden gewisse Voraussetzungen definiert und Bewertungskriterien aufgestellt. Beide haben eine ausführliche Betrachtung erzwungen, indem mehrere Szenarien und verschiedene Cloud-Anbieter betrachtet wurden. Das Ergebnis ist jedes Mal eindeutig: Die sichere Mehrparteienberechnung ist je nach Szenario und Kostenstruktur des Cloud-Anbieters um mindestens den Faktor 28 und maximal den Faktor 2145 preiswerter.

Die Arbeit schließt mit einer Bierdeckelrechnung ab, die stark abstrahiert eine Einordnung des Kostenvergleichs beider Verfahren in Abhängigkeit der Kostenpositionen Betriebskosten (CPU und Speicher) und Netzwerkverkehr ermöglicht. Somit kann ein abstrakter Vergleich beider Verfahren „sofort“ für jeden beliebigen Cloud-Anbieter vollzogen werden.

# Literaturverzeichnis


[1] Bitkom. *Drei von vier Unternehmen nutzen Cloud-Computing*. https://www.bitkom.org/Presse/Presseinformation/Drei-von-vier-Unternehmen-nutzen-Cloud-Computing. Letzter Aufruf am 26.08.2022. Juni 2020.

[2] Adil Bouti. „Homomorphe Verschlüsselung und Cloud-Computing“. de. Diss. University Hagen, 2020. DOI: 10.18445/20200827-130307-0.

[3] BSI (Bundesamt für Sicherheit in der Informationstechnik). *Post-Quanten-Kryptografie*. https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Informationen-und-Empfehlungen/Kryptografie/Post-Quanten-Kryptographie/post-quanten-kryptografie_node.html. Letzter Aufruf am 26.08.2022. Nov. 2017.

[4] Heike Hagemeier. „Kryptografie – heute und zukünftig“. In: *Datenschutz und Datensicherheit - DuD* 43.10 (Aug. 2019), S. 631–635. DOI: 10.1007/s11623-019-1178-3.

[5] Johannes Buchmann. *Einführung in die Kryptographie*. Springer-Verlag GmbH, Apr. 2016. 330 S. ISBN: 9783642397752.

[6] Pascal Paillier. „Public-Key Cryptosystems Based on Composite Degree Residuosity Classes“. In: *Advances in Cryptology — EUROCRYPT '99*. Springer Berlin Heidelberg, 1999, S. 223–238. DOI: 10.1007/3-540-48910-x_16.

[7] R. L. Rivest, A. Shamir und L. Adleman. „A method for obtaining digital signatures and public-key cryptosystems“. In: *Communications of the ACM* 21.2 (Feb. 1978), S. 120–126. DOI: 10.1145/359340.359342.

[8] Danny Dolev, Cynthia Dwork und Moni Naor. „Nonmalleable Cryptography“. In: *SIAM Journal on Computing* 30.2 (Jan. 2000), S. 391–437. DOI: 10.1137/s0097539795291562.

[9] Yevgeniy Dodis, Shai Halevi und Tal Rabin. „A Cryptographic Solution to a Game Theoretic Problem“. In: *Advances in Cryptology — CRYPTO 2000*. Springer Berlin Heidelberg, 2000, S. 112–130. DOI: 10.1007/3-540-44598-6_7.

[10] Craig Gentry. „A Fully Homomorphic Encryption Scheme“. AAI3382729. Diss. Stanford, CA, USA, 2009. ISBN: 9781109444506.

[11] Vinod Vaikuntanathan. „Computing Blindfolded: New Developments in Fully Homomorphic Encryption“. In: *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*. IEEE, Okt. 2011. DOI: 10.1109/focs.2011.98.

[12] Craig Gentry. „Computing arbitrary functions of encrypted data“. In: *Communications of the ACM* 53.3 (März 2010), S. 97–105. DOI: 10.1145/1666420.1666444.

[13] Guillaume Bonnoron. „Somewhat/Fully Homomorphic Encryption: Implementation Progresses and Challenges“. In: *Codes, Cryptology and Information Security*. Hrsg. von Said El Hajji, Abderrahmane Nitaj und El Mamoun Souidi. Cham: Springer International Publishing, 2017, S. 68–82. ISBN: 978-3-319-55589-8.

[14] Yfke Dulek, Christian Schaffner und Florian Speelman. „Quantum Homomorphic Encryption for Polynomial-Sized Circuits“. In: *Advances in Cryptology – CRYPTO 2016*. Hrsg. von Matthew Robshaw und Jonathan Katz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, S. 3–32. ISBN: 978-3-662-53015-3.

[15] Daniele Micciancio und Shafi Goldwasser. *Complexity of Lattice Problems*. Springer US, 2002. DOI: 10.1007/978-1-4615-0897-7.

[16] Oded Regev. „Lattice-Based Cryptography“. In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2006, S. 131–141. DOI: 10.1007/11818175_8.

[17] Cheng Kuan. „Some Complexity Results and Bit Unpredictable for Short Vector Problem“. In: 2013.

[18] Daniele Micciancio. „On the hardness of the shortest vector problem“. Diss. Massachusetts Institute of Technology, 1998.

[19] Miklós Ajtai. „The shortest vector problem in L2 is NP-hard for randomized reductions (extended abstract)“. In: *Proceedings of the thirtieth annual ACM symposium on Theory of computing - STOC '98*. ACM Press, 1998. DOI: 10.1145/276698.276705.

[20] Ishay Haviv und Oded Regev. „Tensor-based hardness of the shortest vector problem to within almost polynomial factors“. In: *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing - STOC '07*. ACM Press, 2007. DOI: 10.1145/1250790.1250859.

[21] Dorit Aharonov und Oded Regev. „Lattice problems in NP ∩ coNP“. In: *Journal of the ACM* 52.5 (Sep. 2005), S. 749–765. DOI: 10.1145/1089023.1089025.

[22] A. K. Lenstra, H. W. Lenstra und L. Lovász. „Factoring polynomials with rational coefficients“. In: *Mathematische Annalen* 261.4 (Dez. 1982), S. 515–534. DOI: 10.1007/bf01457454.

[23] M. Ajtai. „Generating hard instances of lattice problems (extended abstract)“. In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC '96*. ACM Press, 1996. DOI: 10.1145/237814.237838.

[24] D. Micciancio und O. Regev. „Worst-Case to Average-Case Reductions Based on Gaussian Measures“. In: *45th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, 2004. DOI: 10.1109/focs.2004.72.

[25] Oded Regev. „The Learning with Errors Problem (Invited Survey)“. In: *2010 IEEE 25th Annual Conference on Computational Complexity*. IEEE, Juni 2010. DOI: 10.1109/ccc.2010.26.

[26] Oded Goldreich, Shafi Goldwasser und Shai Halevi. „Collision-Free Hashing from Lattice Problems“. In: *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*. Springer Berlin Heidelberg, 2011, S. 30–39. DOI: 10.1007/978-3-642-22670-0_5.

[27] Craig Gentry, Chris Peikert und Vinod Vaikuntanathan. „Trapdoors for hard lattices and new cryptographic constructions“. In: *Proceedings of the fortieth annual ACM symposium on Theory of computing*. ACM, Mai 2008. DOI: 10.1145/1374376.1374407.

[28] Oded Regev. „On lattices, learning with errors, random linear codes, and cryptography“. In: *Journal of the ACM* 56.6 (Sep. 2009), S. 1–40. DOI: 10.1145/1568318.1568324.

[29] Chris Peikert. „Public-key cryptosystems from the worst-case shortest vector problem“. In: *Proceedings of the 41st annual ACM symposium on Symposium on theory of computing - STOC '09*. ACM Press, 2009. DOI: 10.1145/1536414.1536461.

[30] Vadim Lyubashevsky und Daniele Micciancio. „Generalized Compact Knapsacks Are Collision Resistant“. In: *Automata, Languages and Programming*. Springer Berlin Heidelberg, 2006, S. 144–155. DOI: 10.1007/11787006_13.

[31] Damien Stehlé u. a. „Efficient Public Key Encryption Based on Ideal Lattices“. In: *Advances in Cryptology – ASIACRYPT 2009*. Springer Berlin Heidelberg, 2009, S. 617–635. DOI: 10.1007/978-3-642-10366-7_36.

[32] Marten van Dijk u. a. *Fully Homomorphic Encryption over the Integers*. Cryptology ePrint Archive, Paper 2009/616. https://eprint.iacr.org/2009/616. 2009. URL: https://eprint.iacr.org/2009/616.

[33] Junfeng Fan und Frederik Vercauteren. *Somewhat Practical Fully Homomorphic Encryption*. Cryptology ePrint Archive, Paper 2012/144. 2012. URL: https://eprint.iacr.org/2012/144.

[34] Craig Gentry. *A Decade (or So) of Fully Homomorphic Encryption*. Invited talk. 2021.

[35] Zvika Brakerski. „Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP“. In: *Advances in Cryptology – CRYPTO 2012*. Hrsg. von Reihaneh Safavi-Naini und Ran Canetti. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, S. 868–886. ISBN: 978-3-642-32009-5.

[36] Zvika Brakerski, Craig Gentry und Vinod Vaikuntanathan. „(Leveled) Fully Homomorphic Encryption without Bootstrapping“. In: *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*. ITCS '12. Cambridge, Massachusetts: Association for Computing Machinery, 2012, S. 309–325. ISBN: 9781450311151. DOI: 10.1145/2090236.2090262.

[37] Léo Ducas und Daniele Micciancio. „FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second“. In: *Advances in Cryptology – EUROCRYPT 2015*. Hrsg. von Elisabeth Oswald und Marc Fischlin. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, S. 617–640. ISBN: 978-3-662-46800-5.

[38] Chuan Zhao u. a. „Secure Multi-Party Computation: Theory, practice and applications“. In: *Information Sciences* 476 (Feb. 2019), S. 357–372. DOI: 10.1016/j.ins.2018.10.024.

[39] David Evans, Vladimir Kolesnikov und Mike Rosulek. „A Pragmatic Introduction to Secure Multi-Party Computation“. In: *Foundations and Trends® in Privacy and Security* 2.2-3 (2018), S. 70–246. DOI: 10.1561/3300000019.

[40] Manuel Liedel. „Sichere Mehrparteienberechnungen und datenschutzfreundliche Klassifikation auf Basis horizontal partitionierter Datenbanken“. Diss. University of Regensburg, 2012. URL: http://epub.uni-regensburg.de/27630.

[41] Andrew C. Yao. „Protocols for secure computations“. In: *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*. IEEE, Nov. 1982. DOI: 10.1109/sfcs.1982.38.

[42] Michele Ciampi u. a. *Round-Optimal Secure Two-Party Computation from Trapdoor Permutations*. Cryptology ePrint Archive, Paper 2017/920. 2017. URL: https://eprint.iacr.org/2017/920.

[43] Ahmad-Reza Sadeghi, Thomas Schneider und Marcel Winandy. „Token-Based Cloud Computing“. In: *Trust and Trustworthy Computing*. Springer Berlin Heidelberg, 2010, S. 417–429. DOI: 10.1007/978-3-642-13869-0_30.

[44] G. Schaller W. Nüchel. *Nachrichtenverarbeitung Entwurf digitaler Schaltwerke*. Vieweg+Teubner Verlag, März 2013. 223 S. ISBN: 9783322940698.

[45] Tung Chou und Claudio Orlandi. „The Simplest Protocol for Oblivious Transfer“. In: *Progress in Cryptology – LATINCRYPT 2015*. Springer International Publishing, 2015, S. 40–58. DOI: 10.1007/978-3-319-22174-8_3.

[46] Joe Kilian. „Founding crytpography on oblivious transfer“. In: *Proceedings of the twentieth annual ACM symposium on Theory of computing - STOC '88*. ACM Press, 1988. DOI: 10.1145/62212.62215.

[47] Niklas Büscher u. a. „Secure Two-Party Computation in a Quantum World“. In: *Applied Cryptography and Network Security*. Springer International Publishing, 2020, S. 461–480. DOI: 10.1007/978-3-030-57808-4_23.

[48] Donald Beaver, Silvio Micali und Phillip Rogaway. *The Round Complexity of Secure Protocols (Extended Abstract)*. 1990.

[49] Benny Pinkas u. a. „Secure Two-Party Computation Is Practical“. In: *Advances in Cryptology – ASIACRYPT 2009*. Springer Berlin Heidelberg, 2009, S. 250–267. DOI: 10.1007/978-3-642-10366-7_15.

[50] Vladimir Kolesnikov und Thomas Schneider. „Improved Garbled Circuit: Free XOR Gates and Applications“. In: *Automata, Languages and Programming*. Springer Berlin Heidelberg, S. 486–498. DOI: 10.1007/978-3-540-70583-3_40.

[51] Moni Naor, Benny Pinkas und Reuban Sumner. „Privacy preserving auctions and mechanism design“. In: *Proceedings of the 1st ACM conference on Electronic commerce - EC '99*. ACM Press, 1999. DOI: 10.1145/336992.337028.

[52] Vladimir Kolesnikov, Payman Mohassel und Mike Rosulek. „FleXOR: Flexible Garbling for XOR Gates That Beats Free-XOR“. In: *Advances in Cryptology – CRYPTO 2014*. Springer Berlin Heidelberg, 2014, S. 440–457. DOI: 10.1007/978-3-662-44381-1_25.

[53] Samee Zahur, Mike Rosulek und David Evans. „Two Halves Make a Whole“. In: *Advances in Cryptology - EUROCRYPT 2015*. Springer Berlin Heidelberg, 2015, S. 220–250. DOI: 10.1007/978-3-662-46803-6_8.

[54] Claudia Eckert. *IT-Sicherheit*. Gruyter, Walter de GmbH, Aug. 2018. ISBN: 3110551586.

[55] Peter M. Mell und Timothy Grance. *SP 800-145. The NIST Definition of Cloud Computing*. Techn. Ber. Gaithersburg, MD, USA, 2011.

[56] Pierangelo Rosati und Theo Lynn. „Measuring the Business Value of Infrastructure Migration to the Cloud“. In: *Measuring the Business Value of Cloud Computing*. Springer International Publishing, 2020, S. 19–37. DOI: 10.1007/978-3-030-43198-3_2.

[57] Daniele Micciancio und Oded Regev. „Lattice-based Cryptography“. In: *Post-Quantum Cryptography*. Springer Berlin Heidelberg, S. 147–191. DOI: 10.1007/978-3-540-88702-7_5.

[58] Sven Bugiel u. a. „Twin Clouds: Secure Cloud Computing with Low Latency“. In: *Communications and Multimedia Security*. Springer Berlin Heidelberg, 2011, S. 32–44. DOI: 10.1007/978-3-642-24712-5_3.

[59] Dr. Klaus Ruhlig. „Post-Quantum-Kryptografie“. In: *Europäische Sicherheit Technik* (2016).

[60] Kurt Rohloff. *Introducing PALISADE*. https://palisade-crypto.org/wp-content/uploads/2020/08/PALISADE-Introduction-2020-07-24.pdf. letzte Überprüfung am 26.08.2022. Juli 2020.

[61] Ilaria Chillotti u. a. „Faster Fully Homomorphic Encryption: Bootstrapping in Less Than 0.1 Seconds“. In: *Advances in Cryptology – ASIACRYPT 2016*. Hrsg. von Jung Hee Cheon und Tsuyoshi Takagi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, S. 3–33. ISBN: 978-3-662-53887-6.

[62] Xiao Wang, Alex J. Malozemoff und Jonathan Katz. *EMP-toolkit: Efficient MultiParty computation toolkit*. https://github.com/emp-toolkit. Letzter Aufruf am 26.08.2022. 2016.

[63] Thomas P. Ryan. *Sample Size Determination and Power*. John Wiley Sons, Mai 2013. 400 S.

[64] Jung Hee Cheon u. a. „Homomorphic Encryption for Arithmetic of Approximate Numbers“. In: *Advances in Cryptology – ASIACRYPT 2017*. Hrsg. von Tsuyoshi Takagi und Thomas Peyrin. Cham: Springer International Publishing, 2017, S. 409–437. ISBN: 978-3-319-70694-8.

[65] Johannes Buchmann u. a. „Creating Cryptographic Challenges Using Multi-Party Computation“. In: *Proceedings of the 3rd ACM International Workshop on ASIA Public-Key Cryptography - AsiaPKC '16*. ACM Press, 2016. DOI: 10.1145/2898420.2898422.

[66] Xiaoyue Qin, Ruwei Huang und Huifeng Fan. „An Effective NTRU-Based Fully Homomorphic Encryption Scheme“. In: *Mathematical Problems in Engineering* 2021 (Aug. 2021). Hrsg. von Chien Ming Chen, S. 1–9. DOI: 10.1155/2021/9914961.

[67] Wilko Henecka u. a. „TASTY“. In: *Proceedings of the 17th ACM conference on Computer and communications security - CCS '10*. ACM Press, 2010. DOI: 10.1145/1866307.1866358.

[68] Yan Huang u. a. „Faster Secure Two-Party Computation Using Garbled Circuits“. In: *Proceedings of the 20th USENIX Conference on Security*. SEC'11. San Francisco, CA: USENIX Association, 2011, S. 35.

[69] Tibor Jager. „Die Zukunft der Kryptographie“. In: *Datenschutz und Datensicherheit - DuD* 38.7 (Juli 2014), S. 445–451. DOI: 10.1007/s11623-014-0204-8.

[70] O. Goldreich u. a. „Approximating shortest lattice vectors is not harder than approximating closest lattice vectors“. In: *Information Processing Letters* 71.2 (Juli 1999), S. 55–61. DOI: 10.1016/s0020-0190(99)00083-6.

# A Anhang

## A.1 Messergebnisse homomorphe Verschlüsselung

Der folgende Abschnitt zeigt die detaillierten Messergebnisse des Prototyps, der auf der homomorphen, gitterbasierten Verschlüsselung beruht. Alle Durchlaufzeiten sind in Millisekunden angegeben. Der ausgehende Netzwerkverkehr, flüchtige Speicher und der benötigte Festplattenspeicher werden in Kilobyte angegeben, wobei beim Festplattenspeicher lediglich der Speicher der gesamten Installation aufgeführt ist. Die aufgelistete Varianz bezieht sich auf die Populationsvarianz. Die aufgeführten Lesehilfen sind gerundete Werte zur besseren Interpretation der Ergebnisse.

### AES

Die Tabelle 23 zeigt die Messergebnisse der Durchführung des Szenarios AES.

| untersuchte Parameter | | Größe | Lesehilfe |
|---|---|---|---|
| Durchlaufzeit | Mittelwert | 9.548.705,00 ms | 160 m |
| | Standardabweichung | 114.395,10 ms | 2 m |
| | Varianz | 13.086.237.769,97 ms | 21,6 Wochen |
| Speicher | Netzwerkverkehr | - | - |
| | Festplattenspeicher | 142.388 KB | - |
| | Flüchtiger Speicher | 5.557.452 KB | 5,3 GB |

**Tabelle 23:** Messergebnisse des homomorphen Prototypen beim Szenario AES

Die Abbildung 19 zeigt ein Boxplot-Diagramm der Durchlaufzeiten für das Szenario AES.


**Abbildung 19:** Boxplot-Diagramm der Durchlaufzeit des homomorphen Prototyps für das Szenario AES; Werte in Millisekunden (Quelle: Eigene Zeichnung)

### Addition

Die Tabelle 24 zeigt die Messergebnisse der Durchführung des Szenarios Addition.

| untersuchte Parameter | | Größe | Lesehilfe |
|---|---|---|---|
| Durchlaufzeit | Mittelwert | 72.290,87 ms | 1 m 12 s |
| | Standardabweichung | 380,99 ms | - |
| | Varianz | 145.151,46 ms | 2 m 24 s |
| Speicher | Netzwerkverkehr | - | - |
| | Festplattenspeicher | 142.388 KB | - |
| | Flüchtiger Speicher | 5.557.452 KB | 5,3 GB |

**Tabelle 24:** Messergebnisse des homomorphen Prototypen beim Szenario Addition

Die Abbildung 20 zeigt ein Boxplot-Diagramm der Durchlaufzeiten für das Szenario Addition.

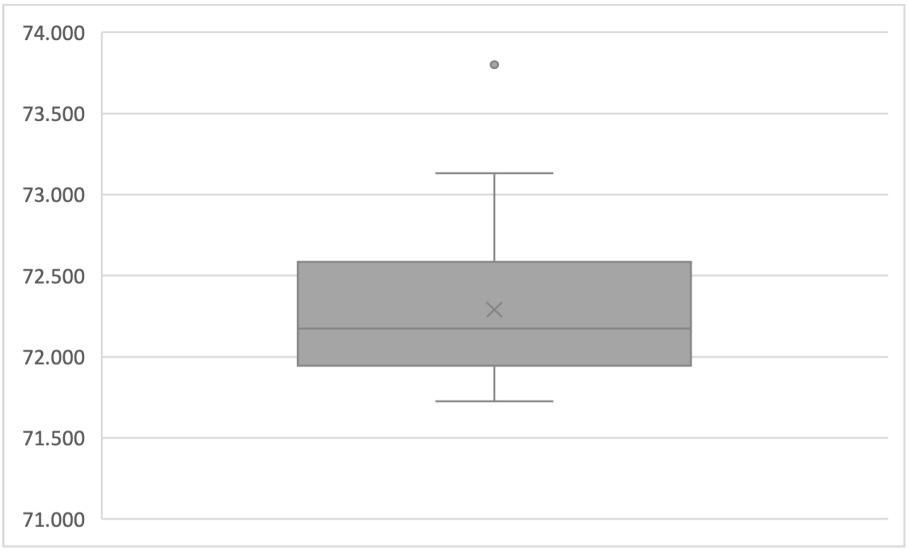



**Abbildung 20:** Boxplot-Diagramm der Durchlaufzeit des homomorphen Prototyps für das Szenario Addition; Werte in Millisekunden (Quelle: Eigene Zeichnung)

### Multiplikation

Die Tabelle 25 zeigt die Messergebnisse der Durchführung des Szenarios Multiplikation.

| untersuchte Parameter | | Größe | Lesehilfe |
|---|---|---|---|
| Durchlaufzeit | Mittelwert | 1.343.027,20 ms | 22 m |
| | Standardabweichung | 9.720,12 ms | 9 s |
| | Varianz | 94.480.788,96 ms | 26,2 h |
| Speicher | Netzwerkverkehr | - | - |
| | Festplattenspeicher | 142.388 KB | - |
| | Flüchtiger Speicher | 5.557.452 KB | 5,3 GB |

**Tabelle 25:** Messergebnisse des homomorphen Prototypen beim Szenario Multiplikation

Die Abbildung 21 zeigt ein Boxplot-Diagramm der Durchlaufzeiten für das Szenario Multiplikation.

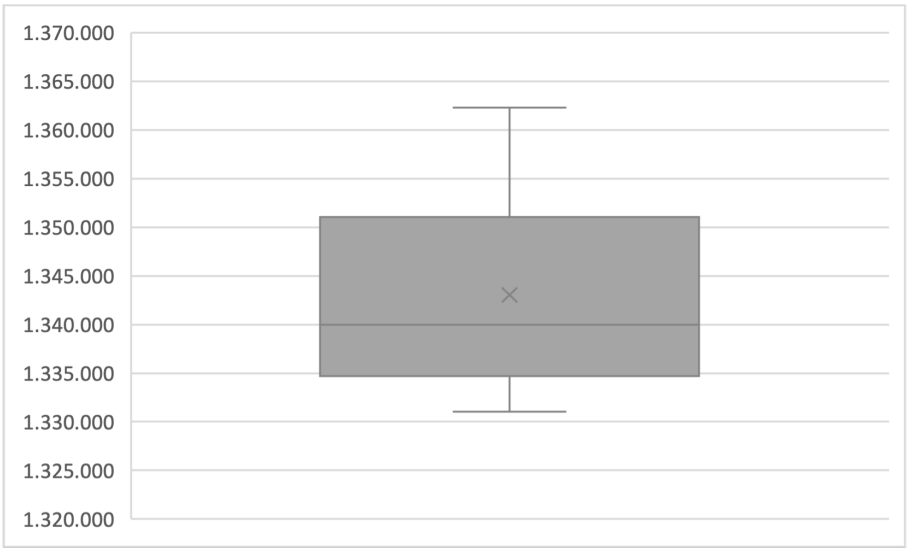



**Abbildung 21:** Boxplot-Diagramm der Durchlaufzeit des homomorphen Prototyps für das Szenario Multiplikation; Werte in Millisekunden (Quelle: Eigene Zeichnung)

## A.2 Messergebnisse Mehrparteienberechnung

Der folgende Abschnitt zeigt die detaillierten Messergebnisse des Prototyps, der auf der sicheren Mehrparteienberechnung beruht. Alle Durchlaufzeiten sind in Millisekunden angegeben. Der ausgehende Netzwerkverkehr, der flüchtige Speicher und der benötigte Festplattenspeicher werden in Kilobyte angegeben, wobei beim Festplattenspeicher lediglich der Speicher der gesamten Installation aufgeführt ist.

### AES

Die Tabelle 26 zeigt die Messergebnisse der Durchführung des Szenarios AES.

Die Abbildung 22 zeigt ein Boxplot-Diagramm der Durchlaufzeiten für das Szenario AES.

| Messergebnis | | Größe |
|---|---|---|
| Durchlaufzeit | Mittelwert | 359,63 ms |
| | Standardabweichung | 11,20 ms |
| | Varianz | 125,88 ms |
| Speicher | Netzwerkverkehr | 5.318 KB |
| | Festplattenspeicher | 40.164 KB |
| | Flüchtiger Speicher | 32.100 KB |

**Tabelle 26:** Messergebnisse des Mehrparteien Prototypen beim Szenario AES

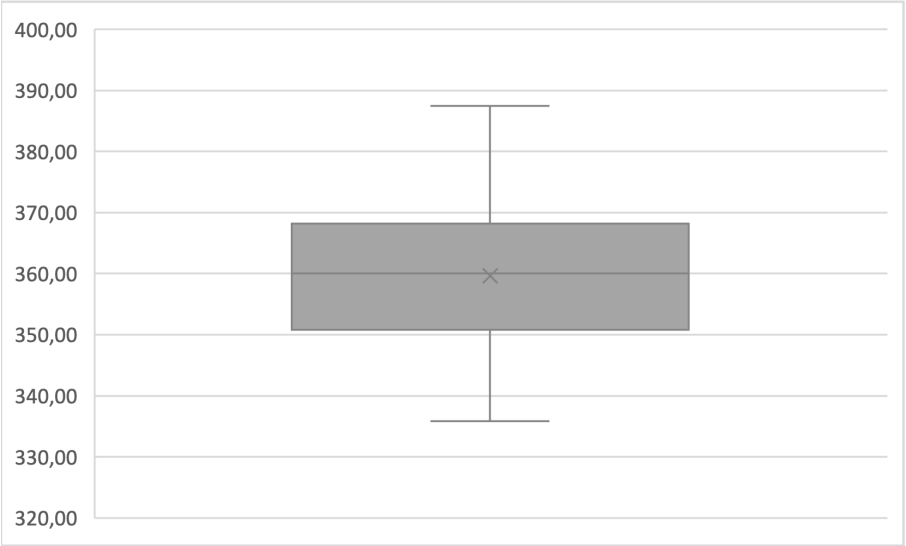



**Abbildung 22:** Boxplot-Diagramm der Durchlaufzeit des Mehrparteien Prototyps für das Szenario AES; Werte in Millisekunden (Quelle: Eigene Zeichnung)

### Addition

Die Tabelle 27 zeigt die Messergebnisse der Durchführung des Szenarios Addition.

Die Abbildung 23 zeigt ein Boxplot-Diagramm der Durchlaufzeiten für das Szenario Addition.

| Messergebnis | | Größe |
|---|---|---|
| Durchlaufzeit | Mittelwert | 60,42 ms |
| | Standardabweichung | 7,83 ms |
| | Varianz | 61,44 ms |
| Speicher | Netzwerkverkehr | 769 KB |
| | Festplattenspeicher | 40.164 KB |
| | Flüchtiger Speicher | 11.200 KB |

**Tabelle 27:** Messergebnisse des Mehrparteien Prototypen beim Szenario Addition

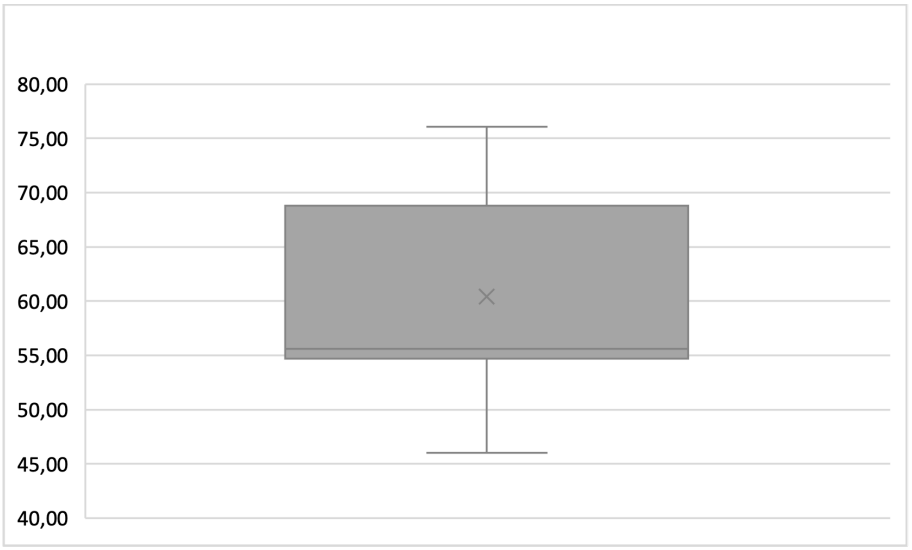



**Abbildung 23:** Boxplot-Diagramm der Durchlaufzeit des Mehrparteien Prototyps für das Szenario Addition; Werte in Millisekunden (Quelle: Eigene Zeichnung)

### Multiplikation

Die Tabelle 28 zeigt die Messergebnisse der Durchführung des Szenarios Multiplikation.

Die Abbildung 24 zeigt ein Boxplot-Diagramm der Durchlaufzeiten für das Szenario Multiplikation.

| Messergebnis | | Größe |
|---|---|---|
| Durchlaufzeit | Mittelwert | 110,38 ms |
| | Standardabweichung | 8,79 ms |
| | Varianz | 77,57 ms |
| Speicher | Netzwerkverkehr | 1.828 KB |
| | Festplattenspeicher | 40.164 KB |
| | Flüchtiger Speicher | 20.400 KB |

**Tabelle 28:** Messergebnisse des Mehrparteien Prototypen beim Szenario Multiplikation

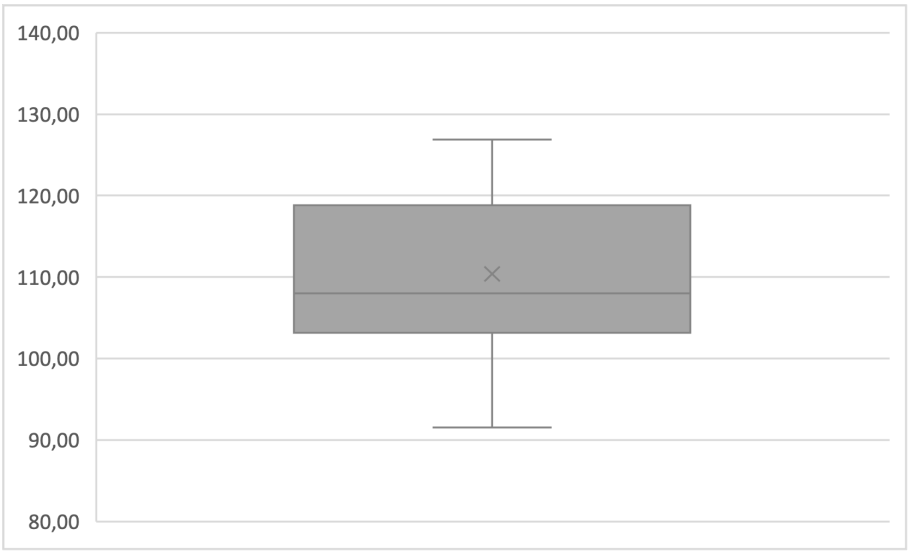



**Abbildung 24:** Boxplot-Diagramm der Durchlaufzeit des Mehrparteien Prototyps für das Szenario Multipliaktion; Werte in Millisekunden (Quelle: Eigene Zeichnung)

# Formelverzeichnis

# Abkürzungsverzeichnis

| | |
|---|---|
| BFV | Brakerski-Fan-Vercauteren Kryptosystem [33]. |
| CKKS | voll-homomorphes Verschlüsselungssystem von Cheon, Kim, Kim and Song [64]. |
| CVP | closest vector Problem. |
| DHKE | Diffie-Hellmann Key Exchange. |
| FHE | voll-homomorphes Verschlüsselungssystem. |
| FHEW | Fastest Homomorphic Encryption in the West Kryptosystem [37]. |
| GC | Garbled Circuits. |
| LHE | leveled homomorphes Verschlüsselungssystem. |
| LWE | learning with errors Problem. |
| OT | Oblivious transfer. |
| RLWE | learning with errors over rings Problem. |
| SHE | Somewhat homomorphe Verschlüsselung. |
| SIS | small integer solution Problem. |
| SVP | shortest vector Problem. |
| TFHE | Fast Fully Homomorphic Encryption over the Torus Kryptosystem [61]. |

## Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe.
Die Stellen der Arbeit, die anderen Quellen im Wortlaut oder dem Sinn nach entnommen wurden, sind durch Angaben der Herkunft kenntlich gemacht. Dies gilt auch für Zeichnungen, Skizzen, bildliche Darstellungen sowie für Quellen aus dem Internet.

Ich erkläre ferner, dass ich die vorliegende Arbeit in keinem anderen Prüfungsverfahren als Prüfungsarbeit eingereicht habe oder einreichen werde.

Die eingereichte schriftliche Fassung entspricht der auf dem elektronischen Wege übermittelten Fassung.

Hamburg, den 28. August 2022 Linus Töbke

# B Thesen

- Die Nutzung von Cloud-Computing wird zukünftig steigen
- Die Vertraulichkeit der Daten bei der Nutzung von Cloud-Computing ist ohne weitere kryptographische Systeme nicht gegeben
- Die Nutzung von homomorpher Verschlüsselung sichert die Vertraulichkeit der Daten bei der Cloud-Nutzung
- Die Nutzung der sicheren Mehrparteienberechnung sichert die Vertraulichkeit der Daten bei der Cloud-Nutzung
- Die genutzten gitterbasierten Verfahren sind quantenresistent und homomorph
- Das genutzte Verfahren bei der sicheren Mehrparteienberechnung ist quantenresistent
- Die gitterbasierten Verfahren sind rechenbasiert
- Das genutzte Verfahren bei der sicheren Mehrparteienberechnung ist netzwerkbasiert
- Die sichere Mehrparteienberechnung ist für den Betrieb in der Cloud preiswerter als die gitterbasierten Verfahren