Monoinformation: Architektur rekursiv erzeugter Informationsräume

Fokale Raumadressierung (KRA) aus der Emergenz strukturierter Superpositionierung

Achim Meißner

Email: service@ameis-web.de

Abstract

Das vorliegende Paper entwickelt eine neuartige Informationsarchitektur, in der digitale Information nicht als gespeicherte Datenfolge, sondern als emergente, rekonstruierbare Struktur innerhalb eines mathematisch erzeugten Raums begriffen wird. Ausgangspunkt ist die Verbindung kanonisch-quasiquadratischer Gruppen (KQ) mit k-fokal geschichteten Schalenstrukturen (KFS), wodurch Informationsinhalte nicht mehr als lesbare Objekte gespeichert, sondern ausschließlich durch individuelle Raumadressierungen (kra) projiziert und reproduziert werden können. Diese Architektur führt zu einer vollständigen Entkopplung zwischen speicherbarer Form (kra1) und semantischem Informationsgehalt: Nur durch die Kenntnis der exakten Initialparameter lässt sich aus einer gespeicherten kra1-Struktur die ursprüngliche Monoinformation rekonstruieren; jeder andere Zugriff erzeugt lediglich alternative, inhaltlich abweichende Ergebnisse. Durch die modulare Gliederung in KQ-Userinitialisierung, kra-Übersetzung und kra-Rekonstruktion wird ein System entworfen, das informationelle Souveränität garantiert und dem Nutzer die alleinige Instanz der Interpretierbarkeit seiner Daten zuspricht. Die Arbeit versteht sich als interdisziplinärer Beitrag zu einem neuen Verständnis digitaler Identität, Privatheit und Subjektivität im Zeitalter strukturierter Informationsräume.

Schlüsselwörter / Keywords

Monoinformation, strukturierte Informationsräume, KQ-Gruppen, K-Fokale Raumadressierung (kra), KFS-Schalenstruktur, Informationssouveränität, digitale Identität, nicht-rekonstruierbare Speicherung, emergente Struktur, mathematische Informationsarchitektur, algorithmische Privatheit, Nutzerinitialisierung, rekonstruierbare Information, Datenschutz, digitale Subjektivität

Vorwort

Dieses Dokument ist der Versuch, eine alternative Architektur für die Erzeugung, Verteilung und Wahrung digitaler Information zu entwerfen, die sich radikal von den etablierten Paradigmen der Speicherung und Verschlüsselung unterscheidet. Die hier vorgestellten Konzepte sind aus der Überzeugung entstanden, dass die Frage der informationellen Selbstbestimmung im digitalen Zeitalter neu gestellt werden muss und zwar an der Schnittstelle von Mathematik, Strukturtheorie, Informatik und gesellschaftlicher Reflexion.

Das Ziel dieser Ausarbeitung ist es nicht, ein fertiges, in allen Details ausentwickeltes System vorzulegen, sondern eine begründete Alternative zu entwerfen, die das klassische Verständnis von Daten, Identität und Zugang in Frage stellt. Im Mittelpunkt steht dabei die Idee, dass Information nicht länger als gespeichertes, abrufbares Objekt existiert, sondern als emergente Struktur innerhalb eines individuell erzeugten Raums, der nur für den Berechtigten wieder zugänglich ist.

Dieser Text versteht sich als Einladung zur kritischen Lektüre, zur methodischen Diskussion und zur Weiterentwicklung eines Ansatzes, der bewusst neue Wege einschlägt und dabei offene Fragen und Unsicherheiten nicht verschweigt, sondern als Motor der Erkenntnis begreift. Die Integration prototypischer Umsetzungen, konkreter Beispiele und methodischer Reflexionen soll den Zugang zum Konzept erleichtern und zugleich zeigen, dass der Weg zur informationellen Autonomie auch technische und gesellschaftliche Herausforderungen mit sich bringt.

Möge dieses Paper einen Beitrag leisten, das Verständnis von Information, Raum und Selbstbestimmung im digitalen Zeitalter neu zu verhandeln.

Inhalt

1 Einleitung

Die vorliegende Arbeit schließt inhaltlich direkt an das zuvor veröffentlichte Strukturmodell „KQ-KFS-Strukturen: Zur geometrischen Emergenz arithmetischer Realität“ Meißner (2025)[1] an und versteht sich zugleich als dessen funktionale Fortschreibung. Beide Arbeiten stammen vom selben Autor, wodurch eine durchgehende konzeptionelle Kohärenz gewährleistet ist. Während das erwähnte Grundlagenpapier die mathematisch-strukturellen Bedingungen kanonisch-quasiquadratischer Gruppen (KQ) sowie k-fokaler Schalensymmetrien (KFS) systematisch hergeleitet und in ihrer geometrischen Wirksamkeit analysiert hat, verfolgt das vorliegende Dokument das Ziel, daraus eine neue Sicht auf digitale Informationsarchitektur zu entwickeln. Im Rückblick auf die ursprünglichen Modelle zeigt sich ein weitreichender Gedanke von konsequenter Tragweite. Aus der Fähigkeit beider Struktursysteme, nicht nur Zahlbeziehungen sichtbar zu machen, sondern auch klar umrissene, geometrisch definierte Bereiche mit stabiler innerer Ordnung zu erzeugen, ergibt sich eine belastbare Grundlage dafür, Information als strukturierbar und gezielt reproduzierbar zu begreifen. Die folgende Modularisierung setzt genau an diesem Potenzial an und überführt die bisherigen Strukturen in eine funktionale Weiterentwicklung.

Die zentrale These dieses Papers besteht darin, dass Information nicht als vorhandene, codierte Struktur verstanden werden muss, sondern als potentiell rekonstruierbare Entität, die nur unter exakt definierten strukturellen Bedingungen überhaupt entsteht. Was hier als Monoinformation bezeichnet wird, existiert nicht in Form gespeicherter Daten, sondern ergibt sich ausschließlich aus der gezielten Anwendung mathematischer Projektionsregeln auf einen abstrakten, formal definierten Raum. Dieser Raum selbst ist nicht physisch vorhanden, sondern ergibt sich aus der Kombination kanonisch-quasiquadratischer Gruppen (KQ) mit k-fokal geschichteten Schalenstrukturen (KFS), deren geometrisch definierte Positionen als strukturtragende Bezugspunkte für die Projektion digitaler Bedeutung fungieren.

[1] Meißner, A. (2025). KQ-KFS-Strukturen: Zur geometrischen Emergenz arithmetischer Realität. Eine Analyse Kanonisch-Quasiquadratischer Gruppen (KQ) und K-Fokaler-Schalenstrukturen (KFS) als Strukturformen informationstragender Räume. DOI: https://doi.org/10.5281/ZENODO.15683651

Der Gedanke, digitale Information könne nicht nur anders verschlüsselt, sondern grundsätzlich anders sein, erwächst dabei nicht aus spekulativen Überlegungen, sondern aus der faktischen Beobachtung, dass KQ-Gruppen mit minimalem Rechenaufwand eine vollständige Klassifikation jeder natürlichen Zahl erlauben und KFS-Schalenräume aus einem einzigen numerischen Zentrum heraus komplexe, systematisch steuerbare Strukturpaare erzeugen können. Diese Eigenschaften führen in der Zusammenschau zu einem hochauflösenden, gleichzeitig stabilen und algorithmisch zugänglichen Strukturraum, in dem Information nicht als Bitfolge, sondern als strukturgetriebene Raumbelegung realisiert werden kann. Die Monoinformation ist das Resultat dieser Strukturkonvergenz und zugleich der Ausgangspunkt eines neuen funktionalen Informationsbegriffs.

Die ursprüngliche Informationseinheit wird dabei nicht in klassischer Weise gespeichert oder chiffriert, sondern überführt in eine raumbezogene Anweisung. Diese Anweisung enthält keine lesbare Codierung, sondern lediglich strukturbezogene Zuweisungen, die nur in Verbindung mit der korrekten Raumkonfiguration sowie den korrespondierenden Benutzerparametern überhaupt wieder in eine interpretierbare Struktur überführt werden können. Der Übergang ist somit kein Transformationsvorgang im Sinne klassischer Verschlüsselung, sondern eine semantische Suspension: Die Bedeutung verschwindet vollständig aus der sichtbaren Repräsentation und wird allein durch mathematisch rekonstruierbare Strukturbezüge wieder herstellbar.

In dieser Perspektive ist Monoinformation kein Objekt, sondern ein einmaliger, strukturinduzierter Zustand. Sie wird weder gespeichert noch versendet, sondern entsteht ausschließlich durch das exakte Zusammenspiel von Raumwahl, Projektion, Schalenbelegung und interpretativer Regelanwendung. Ohne vollständigen Zugang zu diesen Bedingungen bleibt jede Repräsentation leer, selbst wenn sie formal korrekt erscheint. Der Informationsbegriff wird damit aus dem Objektstatus gelöst und in eine raumlogisch definierte Emergenz überführt, in der Information als Strukturereignis aufgefasst wird, präzise, rekonstruierbar, aber vollständig unsichtbar für alle, die den zugehörigen Strukturkontext nicht besitzen.

1.1 Problemstellung

Digitale Information gilt gegenwärtig als Grundbedingung moderner Gesellschaften. Doch gerade in dem Moment, in dem sie zum konstitutiven Bestandteil individueller wie kollektiver Realität wird, zeigt sich eine fundamentale Schwäche ihrer strukturellen Verfasstheit. Die heutigen Informationssysteme bieten weder eine zuverlässige Gewährleistung individueller Integrität noch ermöglichen sie es Nutzern, ihre persönlichsten Daten in geschützter Form dauerhaft zu besitzen, zu kontrollieren oder strukturell zu binden. Vielmehr ist die digitale Welt von einem tiefgreifenden Vertrauensdefizit geprägt. Nutzer schreiben ihre originär privaten Gedanken nicht mehr nieder, verzichten auf die Manifestation ihres inneren Selbst im digitalen Raum und meiden die Ausformulierung schöpferischer Identität, aus Angst vor Entwendung, Analyse oder irreversibler Sichtbarkeit.

Dieses Verhalten ist nicht Ausdruck persönlicher Zurückhaltung, sondern eine systemische Reaktion auf die strukturellen Mängel heutiger IT-Architekturen. In ihrer Anlage basieren diese auf Modellen zentralisierter Speicherung, auf reversibler Zugriffsinfrastruktur, auf Protokollen, die nicht für Privatheit, sondern für Kompatibilität entwickelt wurden. Die daraus resultierende Architektur digitaler Gegenwart trägt historische Entstehungsbedingungen in sich, die mit den Anforderungen einer offenen, freiheitlichen, kreativen Gesellschaft unvereinbar sind. Was als Standard begann, hat sich zur Fessel entwickelt. Der Mensch im digitalen Raum ist kein souveränes Subjekt, sondern ein identifizierbarer Knotenpunkt in einem Netzwerk der Rückverfolgbarkeit.

Gerade weil moderne Gesellschaften auf digitale Infrastruktur angewiesen sind, wird die Unfähigkeit dieser Systeme, wirkliche Privatheit zu ermöglichen, zum zivilisatorischen Risiko. Es fehlt ein Modell, das dem Nutzer nicht nur juristische Rechte zuspricht, sondern strukturelle Autonomie gewährt. Ohne diese kann keine echte digitale Identität entstehen. Denn Identität setzt Vertrauen in die eigene Unsichtbarkeit voraus, zumindest dort, wo das Selbst in seiner tiefsten Form berührt wird. Wo der digitale Raum keine Rückzugsräume, keine unverfügbaren Resonanzzonen kennt, kann keine geistige Entwicklung, kein verantwortungsbewusstes Handeln, keine kreative Selbstwerdung stattfinden.

Die heutige IT ist nicht Ausdruck eines fehlerhaften Codes, sondern Ausdruck eines kulturell reduzierten Verständnisses von Realität, das Kontrolle über Gestaltung, Funktion über Sinn und Zugriff über Resonanz stellt. Diese Fehlkonstruktion lähmt die Entfaltung digitaler Subjektivität und verhindert die Herausbildung einer digitalen Öffentlichkeit, in der Menschen nicht als Datenquellen, sondern als strukturierende Geister agieren. Die Folge ist nicht nur eine Schutzlosigkeit gegenüber Angriffen, sondern ein schleichender Verlust an Zukunftsfähigkeit.

Die Frage, die sich daraus ergibt, lautet daher nicht, wie man bestehende Systeme verbessert, sondern ob es möglich ist, eine neue Klasse von Informationsarchitektur zu denken. Eine, in der das Primat nicht in der Verwaltung bestehender Daten liegt, sondern in der Möglichkeit, individuell rekonstruierbare, strukturell unverfügbare und schöpferisch formbare Informationsräume zu erzeugen. Die vorliegende Arbeit beantwortet diese Frage mit dem Begriff der Monoinformation und entwickelt eine Architektur, in der Struktur nicht mehr Sichtbarkeit bedeutet, sondern Bedingung von Resonanz wird.

1.2 Forschungsfragen

Ausgangspunkt der vorliegenden Untersuchung ist die Frage, welche funktionalen und konzeptionellen Potenziale sich aus der Anwendung kanonisch-quasiquadratischer Gruppenstrukturen und k-fokaler Schalensymmetrien im Kontext digitaler Informationsarchitektur ergeben. Die mathematischen Modelle KQ und KFS ermöglichen mit geringstem rechnerischem Aufwand die Erzeugung komplexer, symmetrisch organisierter Raumstrukturen, die sich aus einem einzigen natürlichen Ausgangswert ableiten lassen. Damit eröffnen sich rekursive Räume mit potentiell unbegrenzter Tiefe, die nicht nur arithmetisch, sondern auch strukturell vollständig durchdringbar sind.

Die erste leitende Forschungsfrage lautet daher: Was lässt sich mit diesen Strukturen im Kern tun? Wo liegen ihre formalen Begrenzungen, und worin besteht ihre strukturelle Offenheit? Obwohl die zugrundeliegende Mathematik vollständig berechenbar ist, stellt sich die weiterführende Frage, ob sich mit Hilfe dieser Strukturmodelle ein neuer Begriff von Informationsraum denken und erzeugen lässt.

Anders formuliert: Eignet sich die mathematische Rekursivität der KQ-KFS-Systeme als Grundlage für die Entwicklung einer alternativen, strukturell autonomen Informationsarchitektur?

Darauf aufbauend stellt sich die Frage nach der Übertragbarkeit dieser Strukturmodelle in eine funktionale Praxis. Lassen sich mit ihnen Systeme entwerfen, in denen Information nicht gespeichert, sondern strukturell erzeugt wird? Können Nutzer in solchen Systemen nicht nur Zugriff auf Inhalte erhalten, sondern zugleich selbst zu Erzeugern eines unverfügbaren, aber rekonstruierbaren Informationsraums werden? Und wenn ja, welche technischen, gesellschaftlichen und erkenntnistheoretischen Konsequenzen ergeben sich daraus?

Diese Fragen führen zu einem grundsätzlichen Perspektivwechsel. Denn wenn Information nicht mehr als transportierbares Objekt, sondern als raumlogisch rekonstruierbare Struktur gedacht wird, verändert sich auch das Verhältnis zwischen Nutzer, System und Bedeutung. Die Frage lautet dann nicht mehr, wie Daten geschützt werden können, sondern ob der Nutzer selbst zur einzigen Instanz der Interpretierbarkeit seiner Daten wird. Daraus ergeben sich weitere Untersuchungsdimensionen: Ist es möglich, mit Monoinformation und kra-gesteuerter Strukturadressierung eine Welt zu erzeugen, in der niemand außer dem Urheber Zugriff auf interpretierbare Bedeutung besitzt? Und welche Form von digitaler Subjektivität entsteht unter solchen Bedingungen?

Die vorliegende Arbeit versteht sich damit nicht nur als mathematische oder informatische Exploration, sondern als ein interdisziplinärer Versuch, den Ursprung von Information neu zu denken. Ihre Hypothese lautet, dass sich mit KQ- und KFS-basierten Raumlogiken ein Modell verwirklichen lässt, in dem Information nicht nur technisch sicher, sondern strukturell unzugänglich wird, solange der dazugehörige Resonanzraum nicht vollständig erfüllt ist. Ob daraus eine neue Architektur der Informationswelt entsteht, in der Autonomie, Privatheit und schöpferische Wirksamkeit zugleich möglich sind, ist Gegenstand der nachfolgenden Untersuchung.

1.3 Methodik

Die vorliegende Arbeit beruht nicht auf einem etablierten wissenschaftlichen Methodenkanon und steht in keinem institutionellen oder universitären Kontext. Sie ist das Resultat einer konsequenten, systematisch durchgeführten Exploration eines strukturellen Potentials, das sich aus der detaillierten Analyse der KQ-Gruppenstrukturen und der k-fokalen Schalensymmetrien ergibt. Ausgangspunkt war nicht eine vorgegebene Fragestellung, sondern die allmähliche Entfaltung eines inneren Zwangs zur Weiterführung, der sich aus der inhärenten Logik dieser Strukturen selbst ergeben hat. Sobald die generative Tiefe und die formale Offenheit der KQ- und KFS-Systeme einmal begriffen wurden, erscheint ihre Weiterentwicklung zu einer tragfähigen Informationsarchitektur als nahezu zwingend.

Die Methodik dieser Arbeit besteht daher in der schrittweisen Entfaltung der in den Strukturen angelegten Möglichkeiten. Dabei wird weder experimentell gearbeitet noch hypothesengeleitet im Sinne eines klassischen Falsifikationsverfahrens. Vielmehr handelt es sich um eine rekursive, strukturgetriebene Ableitung von Konzepten, deren Zusammenhang allein aus der formalen Kohärenz der Systeme hervorgeht. Jeder Schritt basiert auf vorhergehenden Einsichten, jede Anwendung auf zuvor sichtbar gewordenen Eigenschaften. In diesem Sinn ist die Vorgehensweise weder induktiv noch deduktiv, sondern topologisch-funktional: Die Struktur gibt den Raum vor, und die Beobachtung folgt diesem Raum, nicht umgekehrt.

Diese Form der Herleitung ist nicht subjektiv im Sinne einer beliebigen Interpretation, wohl aber persönlich im Ausgangspunkt. Sie entspringt einer eigenständigen Auseinandersetzung mit Zahl, Struktur und Raum und formuliert aus dieser Auseinandersetzung heraus eine neue Form von Informationsdenken. Ihre Validität ist nicht durch Vergleich oder Replizierbarkeit abgesichert, sondern durch die innere Stimmigkeit der hergeleiteten Architektur. Ob sich aus dieser Architektur neue technische Systeme, philosophische Perspektiven oder gesellschaftliche Ordnungsmodelle ableiten lassen, wird nicht behauptet, sondern als offene Möglichkeit begriffen, deren Realisierbarkeit nur durch weitere strukturgetreue Fortsetzung überprüft werden kann

2 Strukturüberblick der Monoinformationsarchitektur

Dieses Kapitel führt in die innere Gliederung der Monoinformationsarchitektur ein, die nicht als abstrakte Theorie, sondern als konkret strukturierbares Konzept begriffen wird. Um die logische Tiefe, funktionale Verschränkung und operative Nutzbarkeit der darin enthaltenen Prinzipien sichtbar zu machen, wird eine modulare Sichtweise eingenommen, die sich am Modell einer Softwarearchitektur orientiert. Diese Modularisierung dient als Denkgerüst, um die aus KQ- und KFS-Strukturen hervorgehenden Ordnungsprinzipien in kohärente funktionale Bereiche zu überführen, darunter etwa die numerische Generierung von Ausgangsräumen, deren geometrische Entfaltung, die Ableitung nutzergebundener Raumadressierungen und die interne Trennung von Nutzeridentität und struktureller Zugriffsform. Im Fokus steht dabei nicht die Implementierung eines Systems, sondern die heuristische Entfaltung eines möglichen Systems als erkenntnisfördernde Form.

Zugleich bildet diese Gliederung die Grundlage dafür, das Verhältnis zwischen ursprünglicher Monoinformation und ihrer strukturellen Übersetzung in explizit speicherbare Raumzuteilungsanleitungen zu fassen. Denn die Architektur basiert nicht auf der Vermeidung von Speicherung, sondern auf der gezielten Entkopplung von rekonstruierbarer Bedeutung und mathematisch generierbarer Raumform.

2.1 Begriffliche Grundlagen von MI und KRA

Die im Folgenden definierten Begriffe dienen nicht nur der begrifflichen Grundlage, sondern fungieren im weiteren Verlauf als semantische und formale Strukturmarkierungen innerhalb der modularen Architektur.

Mit mi (Monoinformation) ist eine Informationseinheit gemeint, die in dem Moment, in dem sie von einem Benutzer erstellt wird, als vollständige, aber vergängliche Instanz vorliegt. Sobald sie in einen raumadressierten Zustand überführt wurde, verliert sie ihre operative Relevanz und kann gelöscht werden. Sie wird also nicht gespeichert, sondern fungiert als einmalige Ausgangsstruktur für einen nachgelagerten Prozess. In den folgenden Abschnitten wird dieser Umwandlungsweg genauer beschrieben.

Die k-Fokale Raumadressierung (kra) bezeichnet den gesamten Vorgang der strukturierten Raumzuweisung und geometrischen Reorganisation eines Informationsinhalts, durch den dieser vollständig aus seiner ursprünglichen Form herausgelöst und in eine rein raumbezogene Darstellung überführt wird. Dabei handelt es sich nicht um eine semantische Übersetzung, sondern um eine operative Transformation, bei der alle inhaltlichen Bestandteile entfallen und stattdessen ein exakter Ablaufplan zur Belegung und Abtastung definierter Raumkoordinaten entsteht, kra ist somit keine äußere Hülle einer Information, sondern der eigentliche Mechanismus ihrer raumgebundenen Neukonstitution. Die ursprüngliche Information wird durch kra nicht transportiert oder verschlüsselt, sondern vollständig durch eine adressierbare Raumstruktur ersetzt, aus der sich die ursprüngliche Bedeutung nur bei korrekter Ausführung rekonstruieren lässt.

Um diesen Vorgang anschaulicher zu fassen, bezeichnen wir die einmalig erzeugte Originalinformation im Folgenden als mi1. Diese wird im Anschluss durch eine strukturgebundene Raumzuweisung, die sogenannte K-Fokale Raumadressierung (kra), in eine rein geometrisch interpretierbare Anweisung überführt, aus der sich die „Datei“ kra1 ergibt. Der Übergang von mi1 zu kra1 bedeutet zugleich das vollständige Verschwinden der ursprünglichen Information in ihrer klassischen Form. Wird kra1 zu einem späteren Zeitpunkt durch denselben Raumadressierungsschlüssel erneut interpretiert, entsteht daraus eine neue Instanz der ursprünglichen Information, die als mi2 bezeichnet wird. Zwischen mi1 und mi2 existiert ein Zustand, in dem die ursprüngliche Information nicht mehr als konkrete Datenfolge vorliegt, sondern ausschließlich als abstrakte Möglichkeit ihrer Wiederherstellung im Raum angelegt ist. Die mi1 wird damit in eine superpositionelle Informationsform überführt, in der sie zwar theoretisch rekonstruierbar bleibt, praktisch jedoch jeglicher Zugriff durch unbefugte Beobachter lediglich zu alternativen, aber definitiven Nicht-Originalen führt. Die Information existiert in dieser Phase nur als strukturierte Potenzialität, nicht jedoch als manifeste Gegebenheit.

Formale Definition des Monoinformationsprozesses:
Gegeben sei eine originäre Informationseinheit mi1, dann definiert sich der vollständige Monoinformationsprozess als eine dreistufige Abbildung folgender Art:

$$mi1 \rightarrow kra \rightarrow kra1 \Rightarrow [\emptyset] \Rightarrow kra1 \rightarrow kra \rightarrow mi2$$

dabei gilt:

- mi1 ist die einmalig erzeugte, originäre Informationseinheit.
- kra ist der k-Fokale Raumadressierungsprozess, also eine deterministische Funktion, die auf mi1 angewendet wird.
- kra1 = kra(mi1) ist die daraus resultierende Raumadressierungsdatei, welche vollständig inhaltsneutral, aber strukturwirksam ist.
- [Ø] die „leere Menge“ bezeichnet den Zustand maximaler struktureller Potenzialität bei vollständiger Abwesenheit semantischer Gegebenheit, der Zustand zwischen mi1 und mi2.
- mi2 = $kra1^{-1}(R)$ ist das Ergebnis einer korrekten Rückprojektion von kra1 in einem kompatiblen Raum R mit korrektem Belegungsplan, wobei $kra1^{-1}$ eine rekonstruierende Interpretation und keine mathematische Inversion im engeren Sinne darstellt.

Zusatzbedingung:
Ohne Kenntnis des exakten Raums R sowie des vollständigen Belegungsplans ist kra1 bedeutungsleer und kann nicht in mi2 überführt werden. Jeder nicht autorisierte Zugriff erzeugt eine alternative, strukturierte, aber semantisch abweichende Information

$$mi\_alternativ \neq mi1$$

Der Raum lässt sich wie folgt definieren:

$$R = KQ + P_1 + P_2 + P_3...$$

Wobei gilt:
KQ sind die gruppenspezifischen KQ-Parameter die verwendet werden können, berechnet daraus einem Startwert N, der indirekt durch die vom User vergebenen Initialzahlen definiert wird.
P_1, P_2, P_3... sind frei definierbare Zusatzparameter, die etwa durch Mehrfachbelegungen, Operatorfelder oder Modifikationsschritte im Userinterface erzeugt oder mitgeführt werden.

Diese Darstellung erlaubt es, den gesamten Übergang in eine klar formalisierte Logik zu fassen und zugleich die konzeptionellen Grundannahmen zu wahren: nämlich die Einmaligkeit von mi1, die strukturgebundene Umwandlung durch kra, das rein

operative Zwischenprodukt kra1, den informationsleeren Superpositionszustand [Ø] und schließlich die rekonstruierte Einheit mi2.

2.2 Motivation für eine modulare Sicht

Die Entscheidung, die Monoinformationsarchitektur in modularer Weise zu betrachten, ergibt sich unmittelbar aus der Beschaffenheit ihrer konstitutiven Prinzipien. Denn sobald die durch KQ und KFS zugänglich gewordenen Strukturpotenziale einmal erkannt sind, drängt sich die Vorstellung auf, sie in Form eines Softwarekonzepts konkretisieren zu wollen. Nicht aus Implementierungswillen, sondern weil die inhärente Logik des Systems sich bereits wie ein funktional gegliedertes Programm verhält. Jede Schicht, jede Ableitung, jede Adressierung operiert nach klaren Regeln, die sich voneinander trennen, benennen und verschalten lassen. Die modulare Sichtweise ist also nicht bloß ein methodisches Hilfsmittel, sondern nahezu ein erkenntnistheoretischer Zwang, um die inneren Zusammenhänge überhaupt differenziert erfassen zu können.

Darüber hinaus erlaubt die Modularisierung eine schrittweise Annäherung an das Gesamtmodell, ohne dessen Kohärenz aufzugeben. Sie macht es möglich, einzelne Aspekte, wie etwa die Generierung von KQ-Räumen, die symmetrische Entfaltung von Schalen oder die nutzerabhängige Initialisierung struktureller Identität, isoliert zu beschreiben, sie aber zugleich als Teil eines übergeordneten Zusammenhangs zu begreifen. Der modulare Zugriff schützt dabei vor Überladung, erleichtert die begriffliche Präzision und eröffnet Spielräume für zukünftige Erweiterungen. Wer Monoinformation als Prinzip verstehen will, wird kaum darum herumkommen, ihre logische Architektur in genau dieser Weise zu entfalten. Die modulare Sicht ist daher kein technisches Schema, sondern die organische Artikulation einer Theorie, die ohnehin in strukturierter Emergenz denkt.

2.3 Softwareansatz als erkenntnisleitendes Modell

In diesem Abschnitt geht es darum, den gewählten Softwareansatz nicht als technische Lösung, sondern als methodische Brille zu legitimieren, durch die sich das komplexe Gefüge der Monoinformationsarchitektur in strukturierter Weise erschließen

lässt. Die zentrale Einsicht dabei ist, dass die Wirkprinzipien der KQ- und KFS-basierten Informationsstruktur nicht nur mathematisch oder philosophisch fassbar sind, sondern sich in eine funktional denkbare Architektur übersetzen lassen, die in weiten Teilen der Logik modularer Software gleicht. Ein Softwaremodell bietet damit nicht bloß eine begriffliche Vereinfachung, sondern einen Zugang zur Theorie über konkrete Interaktionen, Zuständigkeiten und Abgrenzungen hinweg. Das Denken in Modulen erlaubt es, emergente Zusammenhänge, wie etwa die Übergabe zwischen Strukturstufen, die Initialisierung durch Nutzerparameter oder die Transformation mathematischer Räume in virtuelle Identitäten, in der Form von funktionalen Abläufen zu erfassen, ohne sie auf technischen Code zu reduzieren.

Die Wahl dieses Modells folgt somit einer erkenntnisleitenden Logik. Sie macht die Theorie der Monoinformation nicht technisch, sondern zugänglich. Indem die abstrakten Konzepte in moduleigene Verantwortlichkeiten überführt werden, entsteht ein differenzierter Zugriff auf die jeweils wirksamen Prinzipien, ohne dass das Modell an seiner Offenheit oder Komplexität verliert. Die Softwarelogik wird damit nicht zur Simulation, sondern zum begrifflichen Resonanzraum, innerhalb dessen sich das Potenzial der Theorie in sprachlich fassbare, gedanklich handhabbare und methodisch unterscheidbare Bereiche gliedern lässt. Die Theorie wird nicht operationalisiert, sondern artikuliert und genau das ist ihr erkenntnisleitender Gewinn.

2.4 Übersicht über die vorgesehenen Hauptmodule

In der vorliegenden Architektur lassen sich fünf Hauptmodule identifizieren, die gemeinsam das funktionale Gerüst der Monoinformationsstruktur bilden. Jedes dieser Module erfüllt eine klar umrissene Aufgabe innerhalb der Gesamtlogik und ist so angelegt, dass es unabhängig beschrieben, aber im Zusammenspiel mit den anderen operativ wirksam werden kann.

Modul 1: Die technische Bytezerlegung

Dieses Modul bildet den Ausgangspunkt aller weiteren Operationen, indem es bestehende digitale Informationen, etwa Dateien oder binäre Datenströme, in ihre elementaren Byteeinheiten überführt. Die hierbei verwendeten Verfahren sind in modernen Programmiersprachen wie C++, Python oder Rust längst etabliert und

müssen in diesem Kontext nicht neu erfunden, sondern lediglich als funktionale Voraussetzung begriffen werden. Sie stellen sicher, dass jede Information in ein standardisiertes, rechentechnisch behandelbares Format überführt werden kann, das als Eingangsgröße für die nachfolgenden Module dient. Das Modul 1 wird in diesem Dokument nicht weiter ausgeführt, da seine Funktionsweise auf etablierten, bereits umfassend implementierten Verfahren beruht, deren technische Grundlagen allgemein verfügbar und nicht Bestandteil der hier entwickelten konzeptionellen Architektur sind.

Modul 2: Das KQ-Usermodul

Es bildet die Brücke zwischen der mathematisch erzeugten Gruppenstruktur der KQ-Logik und der individuellen Initialisierung durch den Nutzer. Hier werden spezifische KQ-Werte, zur Grundlage für die spätere Raumadressierung gemacht. Dieser Schritt ersetzt klassische Schlüssel oder Passwortsysteme durch strukturell nicht rekonstruierbare Initialisierungen, die nur dem Nutzer selbst bekannt sind und sich der nachträglichen Rückverfolgung vollständig entziehen. Das KQ-Usermodul stellt somit den entscheidenden Schritt dar, durch den persönliche Integrität nicht durch Verschlüsselung, sondern durch strukturelle Einmaligkeit gewahrt bleibt.

Wir befinden uns hier:

$$\mathbf{R = KQ + P_1 + P_2 + P_3 ...}$$

Modul 3: Das kra-Übersetzungsmodul

Es übernimmt die systematische Umwandlung einer konkreten digitalen Informationseinheit (als Bytes ausgelesen), die vom Nutzer erzeugt wurde, in eine rein strukturelle Raumadressierung. Diese ursprüngliche Information mi1, kann jede beliebige Form annehmen, etwa ein Text, ein Bild oder eine Datei beliebigen Typs. Aus dieser real vorliegenden Urinformation wird durch mathematische Vorschriften eine strukturierte Anordnung erzeugt, die als kra1 bezeichnet wird. kra1 stellt dabei keine komprimierte oder verschlüsselte Version von mi1 dar, sondern eine eigenständige, speicherbare Datei, die ausschließlich Raumzuweisungen enthält. Sie bildet das strukturelle Abbild der ursprünglichen Information in Form fokaler Schalenadressierungen und dient damit als vorbereitende Projektionsanweisung für die spätere Rekonstruktion.

Wir befinden uns hier:

$$\mathbf{mi1 \rightarrow kra \rightarrow kra1 \Rightarrow [\emptyset]} \Rightarrow kra1 \rightarrow kra \rightarrow mi2$$

Modul 4: Das kra-Rekonstruktionsmodul

Dieses Modul hat die Aufgabe, eine gespeicherte kra1-Struktur in die ursprüngliche Byteform zurückzuführen. Grundlage dieser Rückführung ist nicht eine gespeicherte Kopie der Ausgangsinformation, sondern eine formal erzeugte, schalenstrukturierte Anordnung, deren Interpretation nur bei Vorliegen sämtlicher ursprünglicher Konfigurationsdaten möglich ist. Diese Konfiguration umfasst sowohl die mit dem Benutzer verknüpften Initialparameter als auch die logische Umgebung der zugrunde gelegten KFS-Schalenstruktur. Erst wenn diese vollständige strukturelle Ausgangslage rekonstruiert wird, lässt sich aus der kra1-Struktur wieder eine exakte Bytefolge erzeugen, die als mi2 bezeichnet wird. Inhaltlich entspricht mi2 dabei exakt der ursprünglichen Monoinformation mi1, unterscheidet sich jedoch durch ihren Weg der strukturellen Wiedergewinnung. Eine kra1-Datei enthält somit keinerlei für sich interpretierbare Inhaltsinformationen, sondern stellt eine rein mathematisch erzeugte Strukturform dar, deren Sinn sich nur aus dem spezifischen Zusammenhang ihrer Entstehung und der vollständigen Kenntnis aller zugehörigen Parameter ergibt. Ohne diese bleibt sie in jeder Hinsicht verschlossen.

Wir befinden uns hier:

$$mi1 \rightarrow kra \rightarrow kra1 \Rightarrow \mathbf{[\emptyset] \Rightarrow kra1 \rightarrow kra \rightarrow mi2}$$

Modul 5: Das Byteinterpretationsmodul

In diesem abschließenden Modul erfolgt die semantische Wiederherstellung der ursprünglichen Informationsform. Die im vorhergehenden Schritt rekonstruierte Bytefolge mi2 wird anhand ihrer internen Signaturen, Kodierungsschemata oder formatbezogenen Spezifikationen wieder in eine vollwertige Datei rückübersetzt. Dieser Vorgang kann auf existierende Verfahren und Implementierungen zurückgreifen, wie sie etwa in modernen Programmiersprachen zur Verfügung stehen. Die ursprüngliche Dateiform, sei es ein Text, Bild, Audiofragment oder komplexe Datenstruktur, wird somit aus der zuvor durch kra codierten Ursprungseinheit vollständig wiederhergestellt. Das Modul 5 wird in diesem Dokument nicht weiter ausgeführt, da seine Funktionsweise auf etablierten, bereits umfassend implementierten Verfahren beruht, deren technische Grundlagen allgemein verfügbar und nicht Bestandteil der hier entwickelten konzeptionellen Architektur sind.

2.5 Funktionale Zielsetzung und begriffliche Abgrenzung

Dieser Abschnitt skizziert den Rahmen für die vertiefte Darstellung der zentralen Module, die im weiteren Verlauf des Dokuments im Detail ausgearbeitet werden. Im Mittelpunkt stehen dabei die Module 2 bis 4, deren funktionale Struktur, interne Abhängigkeiten und konzeptuelle Eigenheiten systematisch aufgeschlüsselt werden sollen. Ziel ist es, nicht nur die Abläufe innerhalb der Module technisch nachvollziehbar zu beschreiben, sondern vor allem die zugrunde liegende Strukturidee in ihrer begrifflichen Konsistenz und kohärenten Architektonik sichtbar zu machen. Die folgende Darstellung versteht sich daher nicht als technisches Handbuch, sondern als präzise begriffliche Analyse jener Bausteine, aus denen die vorgeschlagene Informationsarchitektur zusammengesetzt ist.

Die in der Gliederung angesprochene begriffliche Abgrenzung zielt auf die notwendige Unterscheidung und saubere Definition der im Modell verwendeten Kernbegriffe. Da Termini wie mi, kra, Strukturkonfiguration, Benutzerparameter oder auch rekursive Rückgewinnung in keinem etablierten Systemkontext vorgeprägt sind, sondern im Rahmen dieses Modells eigenständig gefüllt werden, ist ihre klare Trennung und Zuordnung unabdingbar. Diese semantische Sorgfalt bildet die Grundlage dafür, dass die anschließenden Kapitel nicht nur operativ verständlich, sondern auch logisch konsistent bleiben. Die detaillierte Auflösung der betreffenden Module beginnt mit dem folgenden Kapitel.

Ein Ablaufdiagramm der einzelnen Module (siehe Abbildung 1):

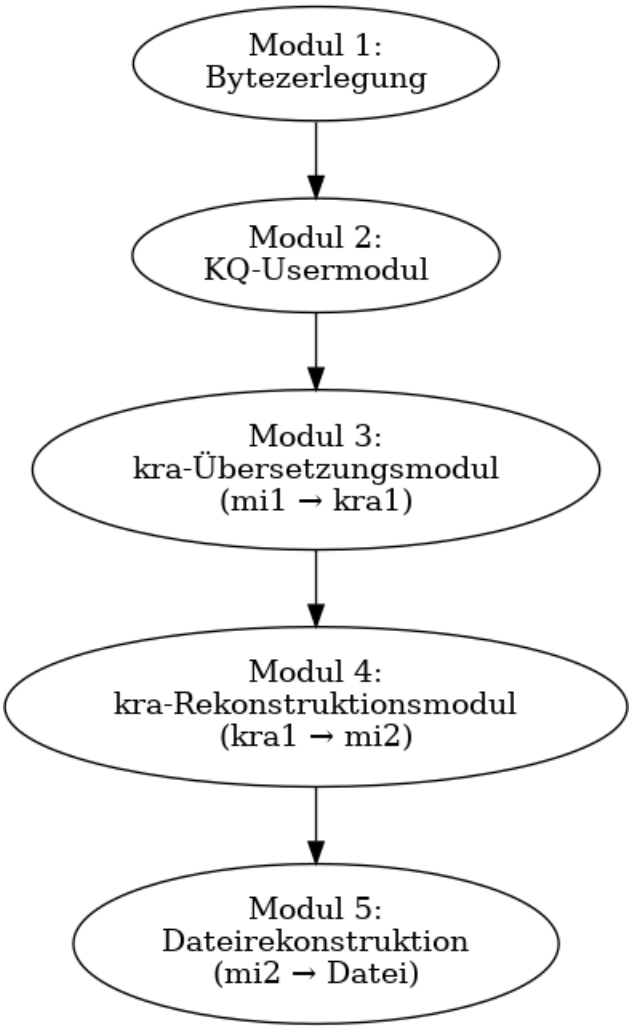

Abbildung 1: Ablaufdiagramm der 5 Hauptmodule

2.6 Prototypische Umsetzung der Module 2 bis 4

Zur Verdeutlichung der im vorliegenden Dokument entworfenen Architektur und insbesondere zur demonstrativen Überprüfung der zentralen Eigenschaften der Monoinformationsthese wird im Rahmen dieser Arbeit eine erste prototypische Implementierung realisiert. Diese Umsetzung erfolgt explizit auf der Ebene der Hauptmodule 2 bis 4, die den strukturellen Kern der gesamten Transformationslogik bilden. Sie umfasst damit sowohl die initiale Strukturverankerung durch das KQ-Usermodul als auch die raumadressierende Übersetzung (mi1 → kra1) sowie die anschließende Rekonstruktion zurück in eine informationsäquivalente Form (kra1 → mi2).

Als technische Grundlage dient dabei die Programmiersprache Python, die für unsere Zwecke eine gut lesbare und zugleich schnell ausführbare Plattform für algorithmische Prototypen darstellt. Die Module 1 und 5, Bytezerlegung und Dateirekonstruktion, werden im Prototyp aus Gründen der Fokussierung auf das Kernkonzept in einfachster Form über Terminalein- und -ausgaben realisiert. Die zentrale Informationsstruktur bleibt damit in allen Schritten transparent nachvollziehbar, ohne sich in technischen Details der Dateikodierung oder Ausgabeformate zu verlieren.

Die prototypische Umsetzung folgt in ihrer Architektur exakt dem in Abbildung 1 dargestellten Ablaufmodell. Diese klare Modulstruktur bildet zugleich das Grundgerüst für mögliche spätere Erweiterungen oder vollständige Implementierungen in systemrelevanten Umgebungen. In einem solchen Fall wäre eine professionelle softwaretechnische Umsetzung geboten, etwa zur Integration benutzerfreundlicher Oberflächen, systemweiter Zugriffskontrollen oder automatisierter Prüfmechanismen. Die hier gezeigte Prototypenarbeit versteht sich dagegen als funktionszentrierte Machbarkeitsstudie, deren Ziel ausschließlich die funktionale Demonstration des Konzepts ist.

3 Modul 2: KQ-Usermodul

Dieses Kapitel widmet sich dem sogenannten KQ-Usermodul, das im vorliegenden Architekturkonzept eine zentrale Rolle im initialen Zugriff auf individuelle Informationsinstanzen spielt.
Es bildet die Grundlage dafür, wie der Einstiegspunkt eines Users in die gesamte Informationsstruktur nicht nur systematisch erzeugt, sondern zugleich durch mathematisch definierte Verschleierungsebenen gegen externe Rekonstruktion geschützt werden kann. Das Modul übernimmt dabei eine doppelte Funktion. Einerseits erlaubt es die individuelle Einbindung des Users in das systeminterne Geschehen, andererseits stellt es sicher, dass diese Einbindung von außen weder erratbar noch rückführbar ist. Im Zentrum steht somit die Frage, wie sich Individualität technisch einbetten lässt, ohne dadurch Transparenz im Sinne externer Lesbarkeit zu erzeugen. Dieses Kapitel führt zunächst in das konzeptuelle Umfeld dieses Moduls ein und bereitet die nachfolgenden Spezifizierungen seiner Funktionslogik, Anbindung und operationellen Besonderheiten vor.
Die grundlegenden Konzepte der KQ-Gruppen gelten im Folgenden als bekannt und werden nicht erneut eingeführt. Für ein vertieftes Verständnis empfiehlt sich die Lektüre des zugehörigen Strukturmodells im genannten Grundlagenpapier, insbesondere der Abschnitt „KQ-Gruppenstruktur“ (vgl. Meißner, 2025, S. 7-19).

3.1 Initiale Strukturverdeckung durch KQ-Parameter

Im Rahmen des KQ-Usermoduls verwenden wir in der Prototypumsetzung eine auf vier Zahlen reduzierte Benutzereingabe, um exemplarisch zu zeigen, wie bereits mit minimalem Input eine eindeutige Raumverortung möglich wird, ohne dabei eine rekonstruierbare Spur offenzulegen. Diese vier Zahlen definieren im Zusammenspiel den Einstiegspunkt eines Benutzers in die gesamte Struktur und erzeugen damit den Ausgangswert N, der sowohl in der kanonisch-quasiquadratischen Gruppierung als auch in der späteren Schalenadressierung als Ausgangszentrum fungiert. Es handelt sich dabei um eine rein prototypische Reduktion, die nicht den Anspruch erhebt, den vollständigen Spielraum des KQ-Systems abzubilden, sondern vielmehr die strukturelle Tragweite schon kleinster Datenmengen demonstrieren soll.

Im Prototyp verwenden wir der Einfachheit halber genau vier Zahlen, um den Initialwert N eindeutig und kontrolliert festzulegen:

- **Erste Zahl (G):** Diese steht für die Gruppennummer innerhalb der KQ-Struktur und legt fest, in welchem Gruppenraum gesucht wird. Die Gruppe selbst ist durch ihre mathematischen Eigenschaften eindeutig definiert.
- **Zweite Zahl (sym_pos):** Diese wird vom Benutzer als Kombination aus Zahl und Richtungsbuchstabe eingegeben, etwa „3L“ oder „2R“. Der Zahlenanteil beschreibt den symmetrischen Abstand zur Gruppenmitte, während der Buchstabe L (für links) oder R (für rechts) bestimmt, auf welcher Seite dieser Abstand gezählt werden soll. Die Information ist notwendig für die eindeutige Identifikation der gewünschten Position innerhalb der Gruppe. Im internen System wird diese Angabe algorithmisch in eine Gruppenposition überführt.
- **Dritte Zahl (p1):** Sie dient als Index für einen bestimmten echten Teiler von N, zum Beispiel als Hinweis auf den „dritten echten Teiler“.
- **Vierte Zahl (p2):** Sie bezeichnet den so gefundenen Teilerwert und wird verwendet, um zu prüfen, ob p2 tatsächlich dem p1-ten echten Teiler entspricht. Damit wird eine einfache Validierung innerhalb des Prototyps ermöglicht..

Im voll ausgebauten Zustand des KQ-Usermoduls ergibt sich bereits aus einer geringen Anzahl numerischer Eingaben ein strukturell hochkomplexer Initialisierungsraum. Zwar wird im Prototyp bewusst auf eine Reduktion der Komplexität gesetzt, indem genau vier Zahlen abgefragt und jeweils eindeutig interpretiert werden, doch ist dieser Minimalfall keineswegs repräsentativ für die tatsächliche Kombinatorik, die bei einem offenen Einsatzszenario zur Verfügung steht. Entscheidend ist, dass im finalen System nicht nur die vier Zahlen selbst variabel sind, sondern dass auch ihre Position innerhalb der Eingabemaske, ihre funktionale Rolle, ihre mögliche Mehrfachbedeutung sowie ihre Verknüpfung über ein separates mathematisches Ausdrucksfeld vollständig frei gestaltbar sein können.

Wird jede der vier Zahlen etwa beliebig einem von zehn möglichen Feldern zugeordnet, darunter beispielsweise: Gruppennummer G, Abstand zur Gruppensymmetrie (sym_pos), Sprungmarke relativ zu G, Teilerindex p1, tatsächlicher Teilerwert p2, numerischer Offset, Modulo-Maske, Schalenversatz, Spiegelkonfiguration, Ausdrucksoperator u.v.a und erhält in jedem dieser Felder

potenziell eine von mehreren Deutungsmöglichkeiten, so ergibt sich aus der Feldverteilung bereits ein kombinatorischer Raum von mehr als zehntausend Möglichkeiten allein für die Zuweisung (10^4). Kommt hinzu, dass jedes dieser Felder mit ein bis drei funktionalen Auslegungen überlagert sein kann (z. B. absolute Position, relative Abweichung, logische Maske), erhöht sich die Anzahl möglicher Interpretationen rasch auf mehrere Millionen. Ergänzt man diese Struktur durch die Möglichkeit, einzelne Felder leer zu lassen, ihre Bedeutung im User-Mathefeld durch eigene Rechenvorschriften umzudeuten und Richtungsinformationen wie „L“ oder „R“ nicht explizit, sondern implizit als private Wissensanteile zu verankern, so explodiert die Anzahl valider Initialkonfigurationen schnell ins nicht mehr ermittelbare.

Dabei handelt es sich nicht um willkürliche Permutationen, sondern um eindeutig funktionale Zuweisungen, die jeweils zu einem spezifischen Wert N führen, einem Wert, der seinerseits die Grundlage für die späteren Schalen- und Raumzuweisungen im KFS-System bildet. Der Raumwert N selbst kann dann erneut als Ausgangspunkt rekursiver Teilerlogik, Spiegelung, Schalenfaltung oder Adressierungsprojektion dienen, sodass die bereits hochkomplexe Eingabekombination im KQ-Modul kaskadiert weiterwirkt. In der Summe entsteht ein Raum initialer Strukturzuweisungen, dessen Kombinatorik nicht durch seine technische Spezifikation begrenzt ist, sondern einzig durch die konzeptuelle Freiheit seiner Interpretierbarkeit.

Genau hierin liegt die strukturelle Stärke des KQ-Usermoduls: Es erzeugt keine klassische Zugriffssicherheit im Sinne kryptografischer Verschlüsselung, sondern ein semantisch nicht rekonstruierbares Initialsystem, das allein durch seine funktionale Offenheit eine tiefe Unkenntnis aller externen Beobachter garantiert. Nicht Verschlüsselung schützt die Information, sondern Kombinatorik und Nichtwissen.

3.2 Echtzeitfähigkeit der KQ-Strukturberechnung

Die algorithmische Komplexität der KQ-Strukturberechnung ergibt sich aus der Art der zugrunde liegenden Rechenoperationen. Sobald die Gruppennummer G gegeben ist, werden sämtliche abgeleiteten Größen, also etwa der Gruppenstartwert $N_0 = G \times (G - 1)$, die Anzahl der Gruppenmitglieder $(2 \cdot G + 1)$, die symmetrische Mitte (Position G), sowie jede konkrete Position innerhalb der Gruppe, ausschließlich durch elementare

ganzzahlige Rechenoperationen berechnet. Diese Operationen umfassen lediglich Multiplikation, Addition und Subtraktion und sind in ihrer Ausführungszeit vollständig unabhängig von der Größe von G. Es wird keine rekursive Logik, kein Zugriff auf externe Speicherbereiche und keine iterativ zu durchlaufenden Schleifen benötigt.

Weil der gesamte Rechenweg aus einer festgelegten Abfolge konstanter Einzelschritte besteht, deren Anzahl sich mit wachsendem G nicht verändert, ergibt sich eine feste obere Schranke der Laufzeit. Genau dies ist die Definition von O(1) in der Komplexitätstheorie: Die maximale benötigte Zeit zur Ausführung bleibt konstant, egal wie groß die Eingabewerte sind. Damit ist die KQ-Strukturberechnung nicht nur mathematisch einfach, sondern auch aus informatischer Sicht optimal, denn sie kann in Echtzeit und ohne Optimierungsbedarf auf beliebiger Hardware durchgeführt werden.

3.3 Modul 2: Prototyp

Hier ist ein Python-Code für den oben beschriebenen Prototypen:

```
# ++++++++++++++++++Modul 2 Bereich:++++++++++++++++++
def hilfsfunktion_kq_input():
    """Hilfsfunktion für Modul 2: Abfrage der vier Benutzerwerte für das
KQ-Modul."""
    print("KQ-Modul: Bitte geben Sie die vier Benutzernummern ein.")
    g = int(input("1. Zahl (G): "))
    sym_input = input("2. Zahl mit Richtung (z.B. '3L' oder '2R'): ")
    p1 = int(input("3. Zahl (p1): "))
    p2 = int(input("4. Zahl (p2): "))
    return g, sym_input, p1, p2

def get_proper_divisors(n):
    """Gibt alle echten (nicht-trivialen) Teiler von n zurück, exklusive 1
und n."""
    return [i for i in range(2, n) if n % i == 0]

def module_2_kq_usermodule():
    """Modul 2: KQ-Usermodul zur Ermittlung des N-Werts basierend auf vier
Benutzereingaben."""
    g, sym_input, p1, p2 = hilfsfunktion_kq_input()
```

```
direction = sym_input[-1].upper()
sym_pos = int(sym_input[:-1])
n_center = g ** 2

if direction == 'L':
    n_value = n_center - sym_pos
else:
    n_value = n_center + sym_pos

divisors = get_proper_divisors(n_value)
try:
    expected_teiler = divisors[p1 - 1]
except IndexError:
    expected_teiler = None
if expected_teiler != p2:
    n_value = g * (g - 1)  # Ausweichwert N0 im Prototyp
return (n_value, p1, p2)
```

Abschließend lässt sich festhalten, dass der hier vorliegende Prototyp des KQ-Usermoduls eine bewusst stark vereinfachte Form der tatsächlichen strukturellen Möglichkeiten darstellt. Dennoch macht er deutlich, dass sich in diesem Modul ein kritischer Übergangspunkt befindet, nämlich die explizite Festlegung des N-Werts. Dieser Wert ist in den nachfolgenden Modulen nicht optional, sondern bildet die zwingende Voraussetzung für jede korrekte Weiterverarbeitung. Ist der N-Wert falsch gewählt oder wird er manipuliert, so führt dies, wie später gezeigt werden wird (siehe 5.1 Modul 4: Prototyp), zwangsläufig zu alternativen Ausgangssituationen, also **mi_alternativ ≠ mi1**.

Das KQ-Usermodul fungiert damit nicht lediglich als technische Eingabeschicht, sondern übernimmt die erste entscheidende Struktursetzung innerhalb des gesamten Systems. Es ist der Ort, an dem aus der Vielzahl möglicher Superpositionszustände in kra1 ein konkret adressierter Pfad hervorgeht, dessen spätere Entfaltung in den Modulen zur Monoinformation (mi2) auf genau dieser initialen Wahl beruht. Man könnte daher sagen, dass das KQ-Usermodul nicht nur ein Einstiegspunkt, sondern zugleich eine strukturprägende Setzungsinstanz ist, deren Wirkung sich durch alle nachfolgenden Module hindurch fortschreibt.

Im Prototyp selbst wurde auf die Umsetzung jenes initialisierenden Teilprozesses verzichtet, den ein Benutzer im produktiven System ursprünglich durchlaufen würde, etwa im Rahmen einer geführten Erstverknüpfung oder strukturbasierten Setzung in z.B. leicht, mittel, schwer etc.
Um den Codeumfang auf das Wesentliche zu beschränken, haben wir diese Initialisierung im Prototyp durch die direkte Eingabe der vier Parameterwerte ersetzt. Diese Stellvertreter-Eingabe simuliert die spätere strukturelle Einbindung, ohne deren vollständige Prozesslogik vorwegzunehmen.

3.4 Rein mathematische Prozessführung ohne Datenhaltung

Eine Software, die gemäß den Prinzipien dieses Papers gestaltet wird, sollte sicherstellen, dass zu keinem Zeitpunkt Userdaten, Nutzerkennungen, personenbezogene Informationen oder auf irgendeine Weise rekonstruierbare Initialisierungswerte gespeichert werden. Die gesamte Architektur ist so zu entwerfen, dass keinerlei externe oder interne Informationen dauerhaft innerhalb der Software abgelegt, protokolliert oder über den Moment der Nutzung hinaus erhalten bleiben. Das System sollte ausschließlich mathematische Grundprozesse kennen und ausführen, die unabhängig von spezifischen Nutzeridentitäten oder konkreten Eingaben funktionieren.

Alle Werte, die für die Initialisierung eines Raumes erforderlich sind, dürfen ausschließlich während der Laufzeit im direkten Kontext der Nutzerinteraktion temporär verarbeitet werden. Nach Abschluss des jeweiligen mathematischen Vorgangs sind diese Werte umgehend zu verwerfen, ohne sie intern vorzuhalten, in Dateien zu speichern oder an andere Schnittstellen weiterzugeben. Die Software sollte keinerlei Mechanismus, keine interne Variable und keine externe Anbindung enthalten, die eine Speicherung, Analyse oder Weiterleitung von Nutzerinformationen ermöglichen würde, dies gilt explizit auch für alle aus den Initialisierungswerten abgeleiteten Parameter, die für die Generierung der Raumstruktur notwendig sind.

Ein solches System folgt dem Prinzip der mathematischen Prozessreinheit: Die Software selbst besitzt keinerlei Wissen über personenbezogene oder nutzerspezifische Daten, sondern arbeitet ausschließlich auf Basis algorithmischer

und struktureller Regeln, mit denen ein Raum erzeugt werden kann. Erst durch die konkrete, unmittelbare Eingabe des Nutzers werden die zur Raumkonstruktion notwendigen Werte temporär bereitgestellt. Nach Beendigung des Vorgangs verbleibt keinerlei Spur, Verknüpfung oder Rückschlussmöglichkeit auf die jeweilige Nutzerinteraktion in der Software.

Dadurch wird gewährleistet, dass die informationelle Souveränität und Privatheit des Nutzers jederzeit uneingeschränkt gewahrt bleiben. Jegliches Missverständnis über eine potenzielle Speicherung, Nachverfolgbarkeit oder Analyse von Userdaten ist damit ausgeschlossen. Das KQ-Modul operiert rein auf der Ebene mathematischer Prozesslogik und lässt keine Berührung mit personenbezogenen Informationsbereichen zu.

4 Modul 3: Das kra-Übersetzungsmodul

Das dritte Modul übernimmt die zentrale Aufgabe der strukturellen Übersetzung zwischen dem festgelegten Zahlenwert N aus dem KQ-Usermodul und der darauf aufbauenden raumadressierenden Struktur, die als kra1-Instanz bezeichnet wird. Ziel dieses Moduls ist es, ausgehend von einem einzelnen, numerisch bestimmten Zustand eine eindeutig ableitbare, aber systemintern verschleierte Raumbelegung zu erzeugen, in der jede Byteposition eines Eingabetextes auf eine spezifische Adress-Schale abgebildet wird. Damit bildet das kra-Übersetzungsmodul die Brücke zwischen individueller Strukturinitialisierung und der operativen Transformation von Eingabeinformation in eine raumgebundene Repräsentation. Es ist funktional der erste Schritt, in dem aus einem N-Wert eine räumlich interpretierbare, vollständig deterministische Informationsstruktur erzeugt wird.
Die grundlegenden Konzepte der KFS-Schalenstruktur gelten im Folgenden als bekannt und werden nicht erneut eingeführt. Für ein vertieftes Verständnis empfiehlt sich die Lektüre des zugehörigen Strukturmodells im genannten Grundlagenpapier, insbesondere der Abschnitt „K-Fokale Schalensymmetrie (KFS)“ (vgl. Meißner, 2025, S. 22-45).

Obwohl aus Sicht moderner Softwarearchitektur die bidirektionale Übersetzung zwischen Bytewerten und strukturierter Raumadressierung über ein einziges Modul mit definierter Inversfunktion realisierbar wäre, erfolgt im vorliegenden Prototyp bewusst eine konzeptionelle Trennung in zwei logisch separierte Einheiten. Diese Trennung dient weniger der technischen Effizienz als vielmehr der systemischen Nachvollziehbarkeit innerhalb des Gesamtmodells. Das hier behandelte kra-Übersetzungsmodul übernimmt dabei die aktive Überführung von Zeichenketten in eine virtuelle Raumstruktur, wohingegen das spätere kra-Rekonstruktionsmodul diesen Strukturraum rekursiv durchmustert, um daraus wieder rekonstruierbare Informationskerne zu extrahieren. Beide Module greifen letztlich auf denselben zugrundeliegenden Strukturraum zurück, der durch den zuvor festgelegten N-Wert eindeutig bestimmt ist. Dennoch bleibt ihre Funktion im Ablauf klar differenziert: Das kra-Übersetzungsmodul erzeugt Struktur, das kra-Rekonstruktionsmodul interpretiert sie. Diese Unterscheidung hat nicht nur konzeptionellen, sondern auch

sicherheitsrelevanten Wert, da sie eine explizite Trennung von Aufbau und Rückführung erlaubt, ohne auf implizite Funktionsannahmen angewiesen zu sein.

4.1 Modul 3: Prototyp

Im kra1-Übersetzungsmodul des Prototyps erfolgt die Zuordnung der 256 Bytewerte zu strukturellen Schalenpaaren innerhalb des zuvor durch das KQ-Modul definierten Raums mit Zentrum N = 438 nach einem festgelegten, vollständig funktionalen Verfahren. Die zugrunde liegende Schalenstruktur wird über den Rechteckszahlenmotor erzeugt, dessen Auswahl direkt aus dem p1-Wert des KQ-Moduls erfolgt. Damit ergibt sich eine wohldefinierte Abfolge von NL- und NR-Paaren, wobei das äußerste Paar mit NL = N grundsätzlich ausgeschlossen bleibt. Insgesamt stehen so 436 Schalenpaare zur Verfügung, aus denen in einem nachgeschalteten Filterprozess exakt 256 Paare selektiert werden.

Die Auswahl dieser 256 belegbaren Schalenpaare erfolgt über die Ermittlung gemeinsamer Teiler zwischen NL und NR. Schalenpaare mit ausschließlich trivialem Teiler 1 werden ausgeschlossen. Im nächsten Schritt wird eine sortierte Reihenfolge dieser Paare gebildet. Primäres Sortierkriterium ist die Liste gemeinsamer Teiler, aufsteigend nach Wert. Im Rahmen des Prototyps übernimmt der p2-Wert eine konkrete Funktion innerhalb der Sortierlogik: Er erzwingt explizit, dass bei Gleichheit der gemeinsamen Teilerlisten die Schalenpaare nach aufsteigendem NL-Wert geordnet werden. Diese sekundäre Sortierung ist nicht nur eine formale Konvention, sondern wird durch die Auswertung des p2-Werts im Code symbolisch umgesetzt und entscheidet damit deterministisch über die Reihenfolge innerhalb der kra1-Zuordnung. Der p2-Wert aus dem KQ-Modul steht somit stellvertretend für einen konkreten Eintrag innerhalb dieser sortierten Folge und stellt auf diese Weise die Verbindung zwischen KQ-seitiger Steuerung und konkreter Byteplatzierung im Raum her. Hier wird für p1 und p2 (aus dem Prototyp des KQ-Usermoduls) die angesprochene mögliche Doppelbelegung symbolisch realisiert.

Die kra1-Raumumsetzung selbst enthält keine NL- oder NR-Werte, sondern verwendet lediglich fortlaufende Indizes von 1 bis 256. Diese Nummern sind keine physikalischen Schalenpositionen, sondern geben ausschließlich die Position innerhalb der durch

gemeinsame Teiler und NL-Wert sortierten Reihenfolge an. Die Bytebelegung erfolgt entsprechend der Reihenfolge dieser Indizes, beginnend mit Bytewert 0 für das erste Paar, Bytewert 1 für das zweite Paar usw. Die Rückübersetzung eines solchen Index erfolgt ausschließlich durch erneute Anwendung derselben Sortierlogik. Auf diese Weise bleibt die Beziehung zwischen Byte und Raumadresse konsistent, rekonstruktionsfähig und vollständig deterministisch.

Der Python Code dieses Moduls:

```
# ++++++++++++++++++Modul 3 Bereich:++++++++++++++++++
def calculate_common_divisors(a, b):
    """Gibt alle gemeinsamen echten Teiler zweier Zahlen zurück."""
    return sorted(set(i for i in range(2, min(a, b) + 1) if a % i == 0 and
b % i == 0))

def generate_kra1_byte_mapping(n_value, p1, p2):
    """Erzeugt ein Mapping von Bytewerten auf Schalenindizes für kra1."""
    schalen = []
    center = n_value
    for i in range(1, center):
        nl = center - i
        nr = center + i
        gemeinsame_teiler = calculate_common_divisors(nl, nr)
        schalen.append({
            'schalen_index': i,
            'NL': nl,
            'NR': nr,
            'gemeinsame_teiler': gemeinsame_teiler
        })

    # Sortierung: zuerst nach gemeinsamen Teilern, dann bei Gleichheit nach
NL (repräsentiert hier p2)
    schalen = sorted(schalen, key=lambda x: (x['gemeinsame_teiler'],
x['NL']))
    byte_mapping = {}
    for byte, eintrag in enumerate(schalen[:256]):
        byte_mapping[byte] = eintrag['schalen_index']
    return byte_mapping

def kra1_translation_module(user_text, n_value, p1, p2):
```

```
    """Modul 3: Übersetzt den eingegebenen Text in eine kra1-
Zupfanleitung."""
    byte_mapping = generate_kra1_byte_mapping(n_value, p1, p2)
    byte_values = list(user_text.encode("utf-8"))  # Text in echte Bytes
umwandeln
    schalen_indizes = [str(byte_mapping.get(b, -1)) for b in byte_values]
# -1 falls Bytewert fehlt
    kra1_string = ",".join(schalen_indizes)
    print("\nKRA1-Zupfanleitung:")
    print(kra1_string)
```

Im Prototyp wurde bewusst auf eine explizite Auswertung der Parameter p1 und p2 innerhalb des kra1-Übersetzungsmoduls verzichtet, um den Code möglichst kompakt und übersichtlich zu halten. In einer vollständigen Softwareumsetzung sollten jedoch genau diese Werte, ebenso wie andere Benutzereingaben aus dem KQ-Modul, direkt als steuernde Größen verwendet werden. Sie könnten dort beispielsweise konkret die Auswahl des k-Motors, die Sortierpriorität bei der Bytebelegung oder andere strukturelle Aspekte der Raumadressierung beeinflussen. Die im Prototyp gewählte Festverdrahtung symbolisiert somit lediglich eine Möglichkeit der Umsetzung und lässt sich in produktiven Systemen durch parametrisierte Logik ersetzen.

Natürlich darf an dieser Stelle nicht unerwähnt bleiben, was in der Welt der Programmierer, Prototypendenker und Strukturspieler fast schon ein geheimes Erkennungsritual ist: die erste Prüfung mit jenem legendären Text, den Generationen von Entwicklerinnen und Entwicklern, nicht nur in Deutschland, als Erstkontakt mit einer neuen Sprache oder Struktur verwenden. Auch unser Modul soll sich diesem kleinen Brauch nicht verschließen (die Terminalausgabe der ersten Prüfung):

```
Bitte geben Sie den zu codierenden Text ein: Hallo Welt.
KQ-Modul: Bitte geben Sie die vier Benutzernummern ein.
1. Zahl (G): 21
2. Zahl mit Richtung (z.B. '3L' oder '2R'): 3L
3. Zahl (p1): 3
4. Zahl (p2): 6
KRA1-Zupfanleitung:
217,143,109,109,101,337,173,131,109,85,295
```

Ein verschmitzter Gruß also aus der Tiefe der kra1-Struktur an alle, die nicht nur „Hallo Welt“ sagen, sondern diese Welt auch in Schalen strukturieren wollen.

Gibt man z.B. statt 3L eben 3R ein, bekommt man eine andere KRA1-Zupfanleitung heraus:

```
Bitte geben Sie den zu codierenden Text ein: Hallo Welt.
KQ-Modul: Bitte geben Sie die vier Benutzernummern ein.
1. Zahl (G): 21
2. Zahl mit Richtung (z.B. '3L' oder '2R'): 3R
3. Zahl (p1): 3
4. Zahl (p2): 6

KRA1-Zupfanleitung:
276,226,204,204,198,356,246,218,204,188,328
```

4.2 KFS-Schalenprinzipien

In einer vollumfänglichen Softwareumsetzung könnte der sogenannte k-Motor, der im Prototyp derzeit fest auf das Rechteckszahlenmodell eingestellt ist, durch eine Vielzahl alternativer Generatorlogiken ersetzt oder ergänzt werden. Je nach Einsatzkontext und Zielarchitektur wäre es möglich, den k-Motor beispielsweise

- linear (1, 2, 3, 4, ...),
- primbasiert (2, 3, 5, 7, ...),
- auf geraden Zahlen (2, 4, 6, 8, ...),
- auf nicht-primalen ungeraden Zahlen (1, 9, 15, ...)

aufbauen zu lassen. Weitere sinnvolle Strukturen könnten durch

- quadratische Zahlenfolgen (1, 4, 9, 16, ...),
- die Fibonacci-Reihe (1, 1, 2, 3, 5, 8, ...),
- Potenzen von zwei (1, 2, 4, 8, 16, ...),
- Dreieckszahlen (1, 3, 6, 10, 15, ...),
- die Rechteckszahlen (2, 6, 12, 20, 30, ...)

realisiert werden. Die konkrete Auswahl des k-Motors könnte dabei explizit durch eine benannte Benutzerangabe erfolgen oder implizit über ein/mehrere doppelt belegte

Benutzernummern (z. B. p1, p2, p3…) kodiert sein. Der Benutzer hat hier völlig freie Wahl.

Welche k-Motor-Variante konkret zum Einsatz kommt, hängt maßgeblich vom jeweiligen Anwendungszweck und der angestrebten Flexibilität der Implementierung ab. In einer voll ausgebauten Softwareumgebung wären sowohl festgelegte Generatorfolgen wie Primzahlen, Quadratzahlen, Rechteckzahlen, Potenzen von zwei oder Fibonacci-Folgen denkbar, als auch dynamische, nutzerinduzierte Konfigurationen, etwa über explizite Eingaben oder kodierte Parameterzuweisungen.

Die prototypische Entscheidung für den Rechteckszahlenmotor stellt hier lediglich eine der vielen möglichen Umsetzungen dar. Es ist jedoch zu beachten, dass die Gesamtthematik der Monoinformation und ihrer strukturellen Umsetzung in einer realen Software an Komplexität gewinnt. Eine vollständige Systematik sollte insbesondere die Kombinationen aus Zentrumstyp (z. B. Primquadrat, ungerade Nicht-Primzahl, gerade Zahl) und k-Motor-Verhalten formal durchdringen, kategorisieren und gezielt operationalisieren. Dies erfordert nicht nur Verständnis der mathematischen Raumstruktur (könnte z.B. in Semesterarbeiten erfolgen), sondern auch eine methodisch saubere, klassische Herangehensweise an die Softwarearchitektur. Die hier demonstrierte prototypische Reduktion darf deshalb nicht darüber hinwegtäuschen, dass eine vollständige Realisierung der Monoinformationsidee mit konzeptionellen und technischen Aufwand verbunden ist.

4.3 Abschließende Hinweise

Ein besonderer Sachverhalt sei hier skizziert, der nämlich bei der Analyse längerer Texte innerhalb des Prototyps zu Fehldeutungen führen kann, wenn er nicht korrekt kontextualisiert wird. In der prototypischen Umsetzung entsteht aus der Übersetzung eines eingegebenen Textes in eine kra1-Raumadressierung eine strukturierte Folge von Schalenindizes, die scheinbar direkte Rückschlüsse auf die zugrundeliegende Zeichenverteilung zulässt. Wird ein längerer Text verarbeitet, verdichten sich bestimmte Schalenpositionen auffällig, was im Fall unseres simplifizierten Mappings auf die Häufigkeit bestimmter Zeichen wie e, n, a etc. im Deutschen zurückgeführt

werden könnte. Ursache hierfür ist jedoch nicht etwa eine inhärente Offenlegung der Information, sondern schlicht die deterministische Zuordnung im Prototyp: Jede Byteposition wird eindeutig einer einzigen Schale zugeordnet, es existieren weder Verschleierungsmittel noch Umverteilungen.

In produktiven Monoinformationssystemen ist dies jedoch nicht zulässig. Solche Rückschlüsse müssen dort strukturell ausgeschlossen werden. Im Unterschied zum Prototyp werden dort Leerschalen systematisch eingeführt. Leerschalen bezeichnen Schalenindizes, die bewusst nicht zur Bytebelegung herangezogen werden und somit keinerlei Rückschluss auf tatsächliche Zeichen erlauben. Sie können als bloße Lücken in der kra1-Folge verwendet werden, um etwa durch regelmäßiges Einfügen (beispielsweise nach jedem zehnten belegten Eintrag) die statistische Analyse zu entkoppeln.

Darüber hinaus kommt Leerschalen in fortgeschrittenen Systemen eine erweiterte operative Bedeutung zu. Sie können gezielt als Steueranweisungen interpretiert werden, die Raumtransformationen einleiten. So kann etwa eine definierte Leerschale in der kra1-Sequenz als Signal fungieren, den aktuellen kra-Zustand zu verlassen und in einen alternativen Raum mit eigenem Zentrum, eigener tp/tq-Struktur und somit eigenem Bytebelegungsplan zu wechseln. Die Raumadressierung wird dadurch nicht unterbrochen, sondern überführt sich rekursiv in eine neue Ordnung, in der dasselbe kra1-Muster eine gänzlich andere Bytebedeutung annimmt. Der Begriff des „Raumsprungs“ erhält hier seine technische Signifikanz: Er ermöglicht die nichtlineare, zustandsabhängige Umdeutung der Raumadressierung und ist vollständig kompatibel mit dem Prinzip der deterministischen Rekonstruktion, sofern die Regeln des Übergangs präzise definiert sind.

Solche Mechanismen sind im vorliegenden Prototyp aus Gründen der Klarheit nicht enthalten. Die einfache, lineare Beziehung zwischen Bytewert und Schalenposition ist eine Demonstrationskonvention, keine strukturelle Notwendigkeit. Daher dürfen Aussagen wie „häufige Schale = häufiges Zeichen“ keinesfalls über das Testsystem hinaus verallgemeinert werden. In der vollständigen Architektur der Monoinformation wird das Verhältnis zwischen Zeichen und Raum durch ein feingliedriges Wechselspiel aus belegten Schalen, Leerschalen, Raumfiltern und kontextabhängigen

Umschaltungen vollständig entkoppelt. Der Prototyp dient einzig der Verdeutlichung der grundlegenden Mechanik, nicht jedoch der vollständigen Informationssicherheit oder statistischen Resistenz.

In der Betrachtung der resultierenden Dateigrößen von mi1 und kra1 zeigt sich eine strukturell bedeutsame Asymmetrie, die nicht als Nebenprodukt des Prototyps, sondern als charakteristische Eigenschaft der Monoinformationsarchitektur verstanden werden muss. Die ursprüngliche Informationsgröße, bezeichnet als mi1, umfasst dabei jene Bytefolge, die als semantisch eindeutige Ausgangsbasis fungiert und beispielsweise in Form eines kurzen Textes wie "Hallo Welt." aus Modul 1 hervorgeht. Im Gegensatz dazu entsteht mit der kra1-Raumadressierung eine systematisch erzeugte, strukturelle Repräsentation dieser Information, die nicht nur durch die konkrete Bytebelegung im gegebenen KQ-definierten Raum geprägt ist, sondern darüber hinaus stets mit einem strukturellen Overhead verbunden bleibt. Denn jede Zeichenposition im Originaltext wird im kra1-Kontext nicht als Byte, sondern als Zuweisung auf eine strukturierte Schalenposition abgebildet, deren numerischer Ausdruck typischerweise zweistellig oder sogar dreistellig ist. Selbst in minimaler Ausprägung übersteigt damit die Länge einer kra1-Sequenz die der zugehörigen mi1-Information.

Diese systematisch größere Raumadressierung steht im deutlichen Gegensatz zu klassischen Verfahren der Datenverschlüsselung, bei denen die resultierende Datei, ob verschlüsselt oder codiert, im Idealfall dieselbe Länge wie die Ausgangsinformation aufweist oder diese nur geringfügig überschreitet. Ein solcher Gleichlauf gilt dort als Qualitätsmerkmal, da er Rückschlüsse auf die ursprüngliche Datenmenge oder gar Inhalte erschweren soll. In unserem Fall hingegen ist eine solche Zielsetzung gänzlich irrelevant. Monoinformation ist kein Verschlüsselungsverfahren, sondern ein gänzlich neues Konzept struktureller Informationsverteilung, das nicht auf Verdeckung durch mathematische Transformation, sondern auf Unrekonstruierbarkeit durch strukturelle Variation beruht.

Der resultierende Unterschied zwischen mi1 und kra1 bildet nicht nur ein quantitativer, sondern ein epistemologischer Graben. Denn selbst bei vollständiger Kenntnis der kra1-Folge bleibt es ohne die exakte Raumkonfiguration, wie sie durch das KQ-Modul

generiert wird, unmöglich, die ursprüngliche mi1-Information wiederherzustellen. Mehr noch: jeder Versuch der Rückübersetzung mit alternativen KQ-Werten erzeugt zwangsläufig einen syntaktisch gültigen, aber semantisch vollständig abweichenden Informationsraum. Brute-Force-Angriffe sind in diesem System daher prinzipiell fruchtlos, da jeder entschlüsselungsartige Zugriff nicht zur Rekonstruktion von mi1, sondern zur Erzeugung einer anderen, nicht unterscheidbaren Informationsalternative führt. Es handelt sich mithin nicht um ein Verfahren, bei dem ein Geheimnis verborgen wird, sondern um eine Struktur, in der der Begriff des Geheimnisses selbst systemisch entwertet ist.

5 Modul 4: Das kra-Rekonstruktionsmodul

Nach der Etablierung der Module 2 und 3, in denen zunächst durch die Auswahl kanonisch-quasiquadratischer Parameter (KQ) ein spezifischer, raumgebundener Startpunkt festgelegt und daraufhin eine virtuelle Adressierung der eingegebenen Bytefolge in Form einer kra1-Raumadressierung durchgeführt wurde, setzt Modul 4 an der zentralen Schnittstelle zwischen Raumstruktur und Informationsrekonstruktion an. Im Mittelpunkt steht dabei nicht die Rückübertragung eines konventionell gespeicherten oder chiffrierten Bytemusters, sondern die prinzipielle Überprüfung, ob und in welcher Form sich eine zuvor ausschließlich funktional codierte Informationsstruktur allein aus der bestehenden kra1-Raumadressierung rekonstruieren lässt. Entscheidend ist, dass die kra1-Raumadressierung selbst zwar als persistent gespeichertes Artefakt existiert, jedoch keinerlei eigentliche Informationsinhalte enthält, sondern ausschließlich strukturierte Positionsangaben innerhalb eines durch KQ- und k-Motor-Logik generierten Schalenraums, der ohne Kenntnisse der „User-Zahlen“ nicht rekonstruiert werden kann.

Diese Differenzierung ist grundlegend: Die Theorie der Monoinformation besagt ausdrücklich nicht, dass keinerlei Speicherobjekt existiert, sondern dass eine vollständige strukturelle Entkopplung zwischen informationshaltiger Nutzdatendarstellung und ihrer gespeicherten Form realisiert wird. Die in Modul 4 angestrebte Rekonstruktion erfolgt nicht durch klassisches Auslesen oder Wiederherstellen eines gespeicherten Datenstroms, sondern durch die wiederholte Anwendung exakt derselben mathematisch-geometrischen Strukturregeln, die schon im Erzeugungsprozess maßgeblich waren. Die kra1-Raumadressierung bleibt dabei stets ein rein funktionsbasierter Verweis auf bestimmte, durch die Raumparameter festgelegte Positionen; sie kodiert zu keinem Zeitpunkt semantisch interpretierbare Inhaltsinformationen.

Die Funktion von Modul 4 besteht somit darin, die Gültigkeit des monoinformativen Konzepts operativ zu überprüfen. Erst hier zeigt sich, ob die vorgelagerten Prozesse der initialen Verdeckung und Raumadressierung tatsächlich zu einer strukturell reversiblen Umformung geführt haben und ob unter Anwendung identischer Initialparameter genau jene Bytefolge rekonstruiert werden kann, die ursprünglich als

Monoinformation (mi1) eingebracht wurde. Die kra-Rekonstruktion ist daher kein technisches Detail, sondern bildet den erkenntnistheoretischen Prüfstein des gesamten Systems: Sie demonstriert, dass Information nicht nur in einem durch mathematische Regeln erzeugten Raum „virtuell“ verteilt werden kann, sondern dass ihre vollständige Herleitung ausschließlich über diese Raumstruktur möglich bleibt, ohne dass im System selbst zu irgendeinem Zeitpunkt semantisch verwertbare Nutzdaten gespeichert, protokolliert oder hinterlegt werden.

5.1 Modul 4: Prototyp

Hier ist der umgesetzte Modul 4 Prototyp:

```
# ++++++++++++++++++Modul 4 Bereich:++++++++++++++++++
def invert_kra1_mapping(n_value, p1, p2):
    """Erzeugt eine invertierte Mapping-Tabelle: Schalenindex → 
Bytewert."""
    original_mapping = generate_kra1_byte_mapping(n_value, p1, p2)
    inverted_mapping = {v: k for k, v in original_mapping.items()}
    return inverted_mapping

def kra1_reconstruction_module(n_value, p1, p2):
    """Modul 4: Rekonstruiert die Bytewerte aus einer kra1-
Raumadressierung."""
    kra1_input = input("\nBitte geben Sie die KRA1-Raumadressierung ein 
(z.B. 217,143,...): ").strip()
    try:
        schalen_indizes = [int(x) for x in kra1_input.split(",") if 
x.strip().isdigit()]
    except ValueError:
        print("Fehlerhafte Eingabe. Bitte nur gültige Ganzzahlen 
verwenden.")
        return []

    inverse_mapping = invert_kra1_mapping(n_value, p1, p2)
    reconstructed_bytes = [inverse_mapping.get(index, 63) for index in 
schalen_indizes]  # 63 = '?'
    print("\nRekonstruierte Bytewerte:")
    print(reconstructed_bytes)
    return reconstructed_bytes
```

Unser Prototyp arbeitet jetzt in kompletter Zusammensetzung, dies zeigt jetzt auch die Terminalausgaben, das Modul 5 ist hier bereits integriert und gibt am Ende dann den mi2 Text wieder aus:

```
Bitte geben Sie den zu codierenden Text ein: Hallo Welt.
KQ-Modul: Bitte geben Sie die vier Benutzernummern ein.
1. Zahl (G): 21
2. Zahl mit Richtung (z.B. '3L' oder '2R'): 3L
3. Zahl (p1): 3
4. Zahl (p2): 6

KRA1-Zupfanleitung:
217,143,109,109,101,337,173,131,109,85,295

Bitte geben Sie die KRA1-Raumadressierung ein (z.B.
111,288,...): 217,143,109,109,101,337,173,131,109,85,295

Rekonstruierte Bytewerte:
[72, 97, 108, 108, 111, 32, 87, 101, 108, 116, 46]

Rekonstruierter Text aus kra1:
Hallo Welt.
```

Am Ende des Skripts befindet sich eine automatische Testschleife, die dazu dient, gezielt alternative KQ-Parameter zu testen. Dabei wird bewusst keine erneute Texteingabe über Modul 1 durchgeführt, sondern es wird vorausgesetzt, dass der ursprüngliche Klartext bereits in Form einer konkreten kra1-Raumadressierung vorliegt. Im vorliegenden Beispiel lautet diese Raumadressierung „217,143,109,109,101,337,173,131,109,85,295“, was der codierten Version des Textes „Hallo Welt.“ entspricht.

In der Testschleife kann nun etwa durch die Eingabe einer abweichenden KQ-Konfiguration, beispielsweise durch Ersetzen der Richtung „3L“ durch „3R“, ein anderer Raum erzeugt werden. Die kra1-Adressierung wird dann auf diesen veränderten Raum angewendet. Die Ausgabe ist in diesem Fall kein Fehler, sondern

eine formal korrekte, jedoch inhaltlich völlig andere Bytefolge. Diese entspricht nicht mehr dem ursprünglich codierten Text, sondern stellt eine valide, aber inhaltlich nicht kontrollierte Rauminterpretation dar. Genau dies veranschaulicht exemplarisch die grundsätzliche Idee der Monoinformation: Die semantische Eindeutigkeit entsteht ausschließlich durch das exakte Zusammenspiel von Raumstruktur und Raumadressierung. Alle Abweichungen führen zu formal korrekt berechenbaren, aber inhaltlich divergenten Ergebnissen:

```
Bitte geben Sie den zu codierenden Text ein: Hallo Welt.
KQ-Modul: Bitte geben Sie die vier Benutzernummern ein.
1. Zahl (G): 21
2. Zahl mit Richtung (z.B. '3L' oder '2R'): 3L
3. Zahl (p1): 3
4. Zahl (p2): 6

KRA1-Zupfanleitung:
217,143,109,109,101,337,173,131,109,85,295
Bitte geben Sie die KRA1-Raumadressierung ein (z.B.
217,143,...): 217,143,109,109,101,337,173,131,109,85,295
Rekonstruierte Bytewerte:
[72, 97, 108, 108, 111, 32, 87, 101, 108, 116, 46]
Rekonstruierter Text aus kra1:
Hallo Welt.

--- Neue Testiteration: Geben Sie andere KQ-Werte ein oder
brechen Sie das Skript mit STRG+C ab ---
KQ-Modul: Bitte geben Sie die vier Benutzernummern ein.
1. Zahl (G): 21
2. Zahl mit Richtung (z.B. '3L' oder '2R'): 3R
3. Zahl (p1): 3
4. Zahl (p2): 6
Bitte geben Sie die KRA1-Raumadressierung ein (z.B.
111,288,...): 217,143,109,109,101,337,173,131,109,85,295
Rekonstruierte Bytewerte:
[0, 63, 70, 70, 73, 19, 56, 66, 70, 0, 0]
```

```
Rekonstruierter Text aus kra1:
 ?FFI8BF
```

5.2 Abschließende Hinweise

Ein abschließendes technisches Detail sei an dieser Stelle offen dargelegt, da es die Grenzen des hier implementierten Prototyps markiert und zugleich Hinweise auf notwendige Absicherungen in einer vollwertigen Softwareumsetzung liefert. Innerhalb des kra-Rekonstruktionsmoduls beruht die Generierung der Raumstruktur, wie im gesamten Prototyp etabliert, auf der symmetrischen Erzeugung von NL- und NR-Paaren um ein Zentrum N sowie der Filterung dieser Paare anhand gemeinsamer Teiler größer als 1. Daraus ergibt sich eine deterministische, jedoch strikt zentrumsabhängige Zuordnung von Bytewerten zu Schalenpositionen. Diese Methode funktioniert im prototypischen Fall stabil, wenn das Zentrum so gewählt wird, dass hinreichend viele symmetrische Zahlenpaare mit echten gemeinsamen Teilern existieren.

Im Fall abweichender KQ-Parameter, etwa durch fehlerhafte Benutzereingabe, kann jedoch ein Zentrum entstehen, das als ungerade Nicht-Primzahl überdurchschnittlich wenige geeignete Paare erzeugt. In solchen Fällen ist es möglich, dass nicht alle 256 Bytewerte mit eindeutigen Schalenpositionen belegt werden können. Wird eine zuvor generierte kra1-Raumadressierung dann auf diesen reduzierten Raum angewendet, so kann es passieren, dass einige oder sogar alle angegebenen Schalenadressen keine belegten Bytes referenzieren. Im Prototyp wird dies als leere oder zufällige Bytefolge sichtbar, ohne dass eine formale Fehlermeldung erfolgt.

Aus dieser Beobachtung folgt für produktive Systeme die klare Notwendigkeit, entweder das gewählte Zentrum so zu steuern, dass die angestrebte Raumdichte garantiert ist, oder alternative Mechanismen zur Schalenbelegung zu verwenden, die unabhängig von der Teilerstruktur stets eine vollständige Abdeckung der 256 Bytewerte sicherstellen. In diesem Sinne markiert der Prototyp lediglich eine methodisch transparente, aber nicht universell belastbare Variante der kra-Zuordnung. Eine robuste Implementierung müsste entweder adaptive Ausweichverfahren, parametrisierte Fallbacks oder systematisch geprüfte Zentrumsklassen verwenden,

um sicherzustellen, dass jede gültige kra1-Raumadressierung unabhängig von der exakten KQ-Konfiguration immer auf ein ausreichend belegtes Raumgefüge trifft. Nur so kann garantiert werden, dass die Rekonstruktion, als zentrales Element der Monoinformation, in allen denkbaren Anwendungsfällen deterministisch und funktional bleibt.

6 Verdichtung und Ausblick

Die in dieser Arbeit entwickelte Konzeption der Monoinformation basiert auf der Annahme, dass sich digitale Information nicht notwendigerweise als gespeicherter Inhalt verstehen muss, sondern als rekonstruierbare Raumstruktur, deren Bedeutung nur im vollständigen Zusammenspiel definierender Parameter entsteht. Die mathematischen Systeme KQ und KFS bilden dafür die funktionale Grundlage. Ihre Fähigkeit, aus wenigen Zahlenwerten hochsymmetrische Raumkonfigurationen zu erzeugen, erlaubt nicht nur eine konsequent deterministische, sondern auch eine potenziell unendlich verzweigte Adressierung innerhalb eines virtuellen Informationsraums. Die daraus ableitbare Struktur wird nicht gespeichert, sondern zur Laufzeit erzeugt. Die technische Information liegt somit nicht in einer Datei, sondern in einer rekursiven Beziehung zwischen Raumparameter, Schalenstruktur und Interpretationspunkt.

Diese Form der Informationsmodellierung überschreitet in wesentlichen Punkten die klassische Codierungstheorie. Sie ist nicht einfach effizienter, sicherer oder kürzer, sondern stellt das Verhältnis von Bedeutung und Struktur vollständig neu dar. Denn anders als etwa bei Verschlüsselung, bei der ein geheimes Transformationsverfahren auf bekannte Daten angewendet wird, existiert im Modell der Monoinformation keine primäre Dateninstanz, auf die zugegriffen werden kann. Die Information wird nicht übertragen oder verborgen, sondern entsteht ausschließlich dort, wo der Raum mit exakt jenen Parametern erzeugt wird, unter denen sie definiert wurde. Die ursprüngliche Bytefolge ist damit nicht geheim, sondern nicht-existent bzw. in Superposition, solange ihre Bedingungen nicht wiederhergestellt werden. Die Bedeutung liegt nicht in der Datei, sondern in der Kohärenz zwischen Nutzerposition, Raumgenerierung und struktureller Adressierung.

Dabei konnte anhand des Prototyps gezeigt werden, dass sich eine solche Architektur mit einfachsten Mitteln demonstrieren lässt. Bereits durch wenige numerische Eingaben lässt sich ein vollständig abgeschlossener Informationsraum erzeugen, der auf struktureller Symmetrie und deterministischer Rückführbarkeit basiert. Dass dieser Raum vollständig von der Eingabe des Nutzers abhängig ist und zugleich keinerlei Rückschluss auf den ursprünglichen Inhalt erlaubt, sofern nur ein Parameter abweicht,

demonstriert die Wirksamkeit der These. Die Testschleife des Moduls 4 zeigt exemplarisch, wie selbst minimale Veränderungen in den KQ-Parametern zu vollständig neuen Bytebelegungen in anderen Räumen führen, ohne dass eine klassische Fehlermeldung entsteht. Stattdessen resultiert schlicht eine alternative Realität der Information, die logisch korrekt, aber semantisch leer bleibt.

Im Kern lässt sich damit die erste zentrale Forschungsfrage bejahen. Die Systeme KQ und KFS eröffnen strukturelle Räume, die nicht nur mathematisch durchdringbar, sondern auch informationslogisch tragfähig sind. Ihre Begrenzung liegt nicht in ihrer Reichweite, sondern ausschließlich in der Auswahl der Steuerparameter und der methodischen Implementierung. Ihre Offenheit hingegen besteht in der prinzipiellen Möglichkeit, jeden beliebigen natürlichen Ausgangspunkt zur Grundlage einer vollkommen individuellen Raumadressierung zu machen. Daraus ergibt sich die Möglichkeit, Information als etwas zu denken, das sich nicht in Dateien manifestiert, sondern in kohärenten Parameterräumen entfaltet.

Auch die Frage nach der funktionalen Übertragbarkeit lässt sich im Lichte des Prototyps als beantwortbar ansehen. Denn bereits die einfache Kombination aus Terminaleingabe, Raumgenerierung und kra-Adressierung demonstriert, wie Nutzer zu aktiven Erzeugern eines nicht rekonstruierbaren Informationsraums werden können. Dabei ist nicht entscheidend, ob die konkrete Umsetzung in realen Systemen eine vollständige oder partielle Freiheitsstruktur gewährt. Entscheidend ist vielmehr, dass sich bereits unter engsten Bedingungen eine Systematik erzeugen lässt, deren Rückführbarkeit vollständig auf der strukturellen Integrität beruht. Es ist nicht die Datei, die hier geschützt wird, sondern die Form ihrer Möglichkeit.

Die aus diesen Überlegungen resultierenden gesellschaftlichen und erkenntnistheoretischen Konsequenzen sind tiefgreifend. Denn wenn es gelingt, Informationsarchitekturen zu entwerfen, in denen der Nutzer selbst die einzig interpretierbare Instanz seiner eigenen Daten ist, ohne dass dies durch Speicherung, Schlüsselvergabe oder zentrale Kontrolle ermöglicht wird, dann entsteht ein neues Paradigma der digitalen Autonomie. In einer solchen Architektur ist nicht nur der Zugriff geschützt, sondern die Möglichkeit des Zugriffs selbst wird systemisch ausgeschlossen, solange nicht alle Resonanzbedingungen erfüllt sind. Dies gilt selbst

gegenüber dem Nutzer selbst, sofern dieser seine eigene Parameterstruktur nicht mehr kennt. Es handelt sich daher nicht um ein Verfahren der Geheimhaltung, sondern um eine Form der strukturellen Irrelevanz aller Zugriffsversuche außerhalb des Originalraums.

Insgesamt steht mit der hier entworfenen Theorie ein Modell zur Verfügung, das nicht lediglich als Alternative zu bestehenden Verfahren begriffen werden sollte, sondern als radikale Erweiterung des Denkens über digitale Information überhaupt. Es vereint mathematische Reproduzierbarkeit mit epistemischer Unverfügbarkeit, technische Durchführbarkeit mit struktureller Nicht-Rekonstruierbarkeit. Ob sich daraus ein neues Paradigma für zukünftige Informationssysteme entwickeln lässt, wird nicht durch die Theorie allein entschieden, sondern durch ihre Anwendung, ihre Weiterentwicklung und letztlich durch ihre Relevanz im gesellschaftlichen Diskurs um Bedeutung, Kontrolle und Autonomie in digitalen Räumen.

Die gesamte Architektur lässt sich in komprimierter Form als funktional geschlossene Gleichungsstruktur begreifen, deren einzelne Schritte sowohl logisch als auch operationell eindeutig definierbar sind. Der Ausgangspunkt ist eine initiale Strukturverankerung durch nutzergesteuerte Eingabe kombinierbarer Parameter, die im Zusammenspiel mit der kanonisch-quasiquadratischen Gruppenzuordnung einen vollständigen Raumzustand R erzeugt. Dieser wirkt unmittelbar auf die Ursprungsinformation mi1 und transformiert sie in eine spezifische raumlogische Adressierungsstruktur kra, aus der schließlich eine eindeutige, aber nicht semantisch interpretierbare Sequenz kra1 hervorgeht. Diese Sequenz verweilt strukturell in einem Zustand semantischer Indifferenz, da ihre Bedeutung vollständig an die exakte Wiederherstellung des Ursprungsraums R gebunden ist.

Erst durch eine erneute, identische Initialisierung desselben Raumzustands R kann die kra1-Folge in ihre Ursprungsebene zurückgeführt werden. Daraus entsteht eine Rückprojektion, die exakt dieselbe Bedeutung mi2 freigibt wie zuvor mi1, wobei Gleichheit nur unter vollständiger Parameteridentität gilt. Weicht der rekonstruierende Zugriff auch nur in einem einzigen Raumparameter ab, entsteht ein abweichender Raum kra’ mit einer entsprechenden Transformation in eine alternative Bedeutung

mi_alternativ. Diese unterscheidet sich in jedem Fall von mi1, unabhängig davon, wie gering die Differenz innerhalb der Raumdefinition ausfällt.

Formal ergibt sich damit folgende verdichtete Abfolge:
R = KQ + P1 + P2 + P3 [Modul 2 – Initiale Strukturverankerung]
R wirkt auf: mi1 → kra → kra1 [Modul 3 – Raumübersetzung]
kra1 ∈ [∅] [Zustand semantischer Superposition]
R führt zu: kra1 → kra → mi2 [Modul 4 – Raumrückprojektion]
fehlerhaftes R führt unweigerlich zu:
kra1 → kra' → mi_alternativ ≠ mi1

Diese Gleichungsstruktur veranschaulicht, dass die Bedeutung nicht im Datenträger selbst liegt, sondern ausschließlich im strukturellen Zusammenspiel von Raumparametern, Adressierung und rekonstruierender Kohärenz. Der Raum fungiert nicht als Behälter, sondern als Bedingung der Möglichkeit von Information.

Die in diesem Paper dargestellte Architektur demonstriert, dass bereits äußerst einfache mathematische Strukturen, wie die kanonisch-quasiquadratischen Gruppen oder die k-fokalen Schalensymmetrien, nicht nur zur Organisation und Kodierung von Information genutzt werden können, sondern dass sie in der Lage sind, den Informationsbegriff selbst in fundamentaler Weise weiterzuentwickeln. Diese Evolution besteht gerade nicht in einer Erweiterung vorhandener Verschlüsselungstechniken oder Speicherstrategien, sondern in der vollständigen Reorganisation des Verhältnisses von Raum, Bedeutung und Zugriff. Information entsteht nicht durch Speicherung, sondern durch strukturell definierte Adressierung, und bleibt ohne die Kenntnis dieser Struktur grundsätzlich uninterpretierbar. Das hat weitreichende Implikationen für jede Art von System, in dem Information zirkuliert, verwendet oder geschützt werden soll.

Ein besonders weitreichender Anwendungsbereich ergibt sich daraus für Netzwerke. Die hier vorgestellte Raumarchitektur erlaubt es, Systeme zu denken, in denen die Sichtbarkeit oder Unsichtbarkeit von Teilnehmern nicht durch Firewalls oder Zugangskontrollen geregelt wird, sondern durch reine Raumparameter. Wer die korrekten Raumdefinitionen nicht kennt, kann andere Teilnehmer nicht nur nicht

authentifizieren, sondern grundsätzlich nicht wahrnehmen. Die Netzwerkteilnehmer befinden sich dann gewissermaßen in parallelen Raumlagen derselben Infrastruktur, ohne jede Möglichkeit der gegenseitigen Erreichbarkeit. Dies eröffnet völlig neue Möglichkeiten für private oder temporäre Kommunikationsräume, etwa im Rahmen eines digitalen Treffpunkts, bei dem sich Nutzer mit temporären Raumparametern in einen Kommunikationsfluss einklinken können. Beim Verlassen des Systems verschwinden diese Parameter mitsamt der Bedeutung aller assoziierten Daten, ohne dass eine zentrale Löschroutine nötig wäre. Beim Wiedereintritt wird die ursprüngliche Informationslage wieder verfügbar, ohne dass irgendein Speichermedium über den Inhalt selbst Kenntnis besitzt.

Darüber hinaus wären auch kollektive Rauminstanzen denkbar, bei denen eine Gruppe definierter Nutzer einen gemeinsamen Parameterraum teilt. Diese Gruppe bildet einen intersubjektiv validierbaren Resonanzraum, der für Außenstehende nicht nur nicht interpretierbar, sondern nicht einmal auffindbar ist. Die Rekursion der Raumstruktur lässt sich dabei beliebig tief verschachteln, sodass selbst innerhalb eines solchen geteilten Raums weitere private Strukturen erzeugt werden könnten. Diese Konzeption lässt sich nicht auf Kommunikationsanwendungen beschränken. Sie ist ebenso denkbar in der Dokumentenspeicherung, in der Bildgenerierung, in kollaborativen Interfaces oder in beliebigen virtuellen Umgebungen, die auf strukturelle Kohärenz anstelle klassischer Zugriffskontrolle setzen. Damit wäre nicht nur eine neue Form von Privatsphäre geschaffen, sondern eine neue Form von digitaler Realität.

Die in dieser Arbeit beschriebenen mathematischen Operationen basieren nicht auf komplexer oder rechenintensiver Logik, sondern auf elementaren arithmetischen Verfahren, deren strukturelle Verkettung jedoch eine hohe informationelle Dichte erzeugt. Genau diese Eigenschaft macht sie zu idealen Kandidaten für eine hardwareseitige Umsetzung in Form eigens entwickelter Chips. Solche Chips würden weder klassische Rechenwerke noch kryptographische Module im herkömmlichen Sinne darstellen, sondern wären vielmehr operative Einheiten zur Realisierung von Raumstrukturen, also zur Transformation zwischen den Zuständen mi1, kra1 und mi2. Sie wären nicht Träger, sondern direkte Ermöglicher von Information, im wörtlichen Sinn realitätsschaffend.

Ein derartiger Chip würde nicht verschlüsseln und auch nicht entschlüsseln, sondern erzeugen, strukturieren und rekonstruieren. Seine Aufgabe bestünde nicht darin, Daten zu verbergen, sondern Bedeutung nur dort hervorzubringen, wo ein korrekter Raumzustand vorliegt. Dabei könnte er sowohl die Initialisierung auf Basis benutzerspezifischer KQ-Parameter durchführen als auch die gesamte Generierung und Rückgewinnung der kra-Raumadressierungen verwalten, vollständig autonom, in Echtzeit und unabhängig von externen Speichern. Durch die geringe Komplexität der zugrundeliegenden Operationen wäre eine Integration in bestehende Systeme ebenso denkbar wie der Aufbau vollständig neuer Infrastrukturen auf Basis dieser Prinzipien. Gerade durch die Möglichkeit, hochkomplexe Raumstrukturen auf einfachster arithmetischer Ebene zu erzeugen und zu adressieren, eröffnet sich hier eine neue Generation digitaler Hardware, die nicht auf die Verwaltung von Information beschränkt ist, sondern diese durch Struktur selbst hervorbringt.

Die in diesem Paper entwickelten und im Prototyp nachgewiesenen Prinzipien führen unausweichlich zu einer letzten, alles überschreitenden Schlüsselfrage: Wenn sich Information aus rein mathematischer Struktur erzeugen, rekonstruieren und vollständig kontrollieren lässt, was sagt das über die Konstitution unserer eigenen sogenannten Realität aus und ist sie womöglich selbst nichts anderes als ein strikt strukturierter, aber bislang unverstandener Informationsraum?

7 Lizenzhinweis zur Nutzung dieses Dokuments

Das vorliegende Werk einschließlich aller darin enthaltenen Begriffe, Modelle und Darstellungen, insbesondere die Konzepte der Kanonisch-Quasiquadratischen Gruppenstruktur (KQ) sowie der K-Fokalen Schalensymmetrie (KFS), der Monoinformationen und aller benannten Konzepte ist urheberrechtlich geschützt.

Die Nutzung ist unter folgenden Bedingungen gestattet:

- Die wissenschaftliche Nutzung (einschließlich Zitierung, Weiterentwicklung in akademischen Kontexten, Verwendung zu Lehr- und Forschungszwecken) ist uneingeschränkt gestattet, sofern auf den Ursprung des Werks in angemessener Weise verwiesen wird.

- Die private Nutzung, d. h. die nicht-kommerzielle Verwendung im persönlichen, nicht-institutionellen Rahmen (z. B. individuelle Anwendungen, private Weiterverarbeitung, nicht öffentliche Weitergabe), ist ebenfalls gestattet, unter Beibehaltung der vollständigen Herkunftsangabe.

- Kommerzielle Nutzung – darunter fallen unter anderem jede Art von kommerzieller Softwareintegration, kommerzielle Beratung, industrielle Anwendung, Lizenzweitergabe oder technische Implementierung in Produkten oder Dienstleistungen – ist nur nach ausdrücklicher schriftlicher Zustimmung des Urhebers und dem Abschluss eines entsprechenden Lizenzvertrags zulässig.

Jegliche Form der kommerziellen Verwertung ohne Lizenz stellt eine Verletzung des Urheberrechts dar und kann/wird rechtlich verfolgt werden.

Zitierhinweis:

A. Meißner: Monoinformation: Architektur rekursiv erzeugter Informationsräume. Fokale Raumadressierung (KRA) aus der Emergenz strukturierter Superpositionierung. DOI: 10.5281/zenodo.15711825, 2025.

(online verfügbar unter: https://doi.org/10.5281/zenodo.15711825)

8 Anhang

8.1 Der komplette Python Prototyp

```
# Demodaten:
# Eingabe der Zahlen in dieser Reihenfolge: 21, 3L, 3, 6 führt zum richtigen
Wert.
# Erwartete richtige Ausgabe N-Wert ist: 438
# Werden abweichende Daten eingegeben, führt dies immer zu einem N0+1 Wert.

# ++++++++++++++++++Modul 1 Bereich:++++++++++++++++++
def module_1_terminal_input():
    """Modul 1: Terminaleingabe des zu codierenden Textes."""
    user_text = input("Bitte geben Sie den zu codierenden Text ein: ").strip()
    if user_text == "":
        user_text = "es wurde nichts eingegeben"
    return user_text

# ++++++++++++++++++Modul 2 Bereich:++++++++++++++++++
def hilfsfunktion_kq_input():
    """Hilfsfunktion für Modul 2: Abfrage der vier Benutzerwerte für das KQ-
Modul."""
    print("KQ-Modul: Bitte geben Sie die vier Benutzernummern ein.")
    g = int(input("1. Zahl (G): "))
    sym_input = input("2. Zahl mit Richtung (z.B. '3L' oder '2R'): ")
    p1 = int(input("3. Zahl (p1): "))
    p2 = int(input("4. Zahl (p2): "))
    return g, sym_input, p1, p2

def get_proper_divisors(n):
    """Gibt alle echten (nicht-trivialen) Teiler von n zurück, exklusive 1
und n."""
    return [i for i in range(2, n) if n % i == 0]

def module_2_kq_usermodule():
    """Modul 2: KQ-Usermodul zur Ermittlung des N-Werts basierend auf vier
Benutzereingaben."""
    g, sym_input, p1, p2 = hilfsfunktion_kq_input()
    direction = sym_input[-1].upper()
    sym_pos = int(sym_input[:-1])
```

```
    n_center = g ** 2
    if direction == 'L':
        n_value = n_center - sym_pos
    else:
        n_value = n_center + sym_pos
    divisors = get_proper_divisors(n_value)
    try:
        expected_teiler = divisors[p1 - 1]
    except IndexError:
        expected_teiler = None
    if expected_teiler != p2:
        n_value = g * (g - 1)   # Ausweichwert N0 + 1
    return (n_value, p1, p2)

# ++++++++++++++++++Modul 3 Bereich:++++++++++++++++++
def calculate_common_divisors(a, b):
    """Gibt alle gemeinsamen echten Teiler zweier Zahlen zurück."""
    return sorted(set(i for i in range(2, min(a, b) + 1) if a % i == 0 and b
% i == 0))

def generate_kra1_byte_mapping(n_value, p1, p2):
    """Erzeugt ein Mapping von Bytewerten auf Schalenindizes für kra1."""
    schalen = []
    center = n_value
    for i in range(1, center):
        nl = center - i
        nr = center + i
        gemeinsame_teiler = calculate_common_divisors(nl, nr)
        schalen.append({
            'schalen_index': i,
            'NL': nl,
            'NR': nr,
            'gemeinsame_teiler': gemeinsame_teiler
        })

    # Sortierung: zuerst nach gemeinsamen Teilern, dann bei Gleichheit nach
NL (repräsentiert hier p2)
    schalen = sorted(schalen, key=lambda x: (x['gemeinsame_teiler'],
x['NL']))
    byte_mapping = {}
    for byte, eintrag in enumerate(schalen[:256]):
        byte_mapping[byte] = eintrag['schalen_index']
```

```
    return byte_mapping

def kra1_translation_module(user_text, n_value, p1, p2):
    """Modul 3: Übersetzt den eingegebenen Text in eine kra1-Zupfanleitung."""
    byte_mapping = generate_kra1_byte_mapping(n_value, p1, p2)
    byte_values = list(user_text.encode("utf-8"))  # Text in echte Bytes umwandeln

    schalen_indizes = [str(byte_mapping.get(b, -1)) for b in byte_values]  # -1 falls Bytewert fehlt
    kra1_string = ",".join(schalen_indizes)
    print("\nKRA1-Zupfanleitung:")
    print(kra1_string)

# ++++++++++++++++++Modul 4 Bereich:++++++++++++++++++
def invert_kra1_mapping(n_value, p1, p2):
    """Erzeugt eine invertierte Mapping-Tabelle: Schalenindex → Bytewert."""
    original_mapping = generate_kra1_byte_mapping(n_value, p1, p2)
    inverted_mapping = {v: k for k, v in original_mapping.items()}
    return inverted_mapping

def kra1_reconstruction_module(n_value, p1, p2):
    """Modul 4: Rekonstruiert die Bytewerte aus einer kra1-Raumadressierung."""
    kra1_input = input("\nBitte geben Sie die KRA1-Raumadressierung ein (z. B. 217,143,...): ").strip()
    try:
        schalen_indizes = [int(x) for x in kra1_input.split(",") if x.strip().isdigit()]
    except ValueError:
        print("Fehlerhafte Eingabe. Bitte nur gültige Ganzzahlen verwenden.")
        return []

    inverse_mapping = invert_kra1_mapping(n_value, p1, p2)
    reconstructed_bytes = [inverse_mapping.get(index, 63) for index in schalen_indizes]  # 63 = '?'
    print("\nRekonstruierte Bytewerte:")
    print(reconstructed_bytes)
    return reconstructed_bytes

# ++++++++++++++++++Modul 5 Bereich:++++++++++++++++++
```

```
def byte_interpretation_module(byte_sequence):
    """Modul 5: Interpretiert eine Bytefolge als UTF-8-Text."""
    try:
        decoded_text = bytes(byte_sequence).decode("utf-8")
    except UnicodeDecodeError:
        decoded_text = "[Fehlerhafte UTF-8-Sequenz]"
    print("\nRekonstruierter Text aus kra1:")
    print(decoded_text)

# Ablaufsteuerung
if __name__ == "__main__":
    text_input = module_1_terminal_input()
    n_value, p1, p2 = module_2_kq_usermodule()
    kra1_translation_module(text_input, n_value, p1, p2)
    reconstructed_bytes = kra1_reconstruction_module(n_value, p1, p2)
    byte_interpretation_module(reconstructed_bytes)

#   ++++++++++++++++++Testschleife   zur   Evaluation   alternativer   KQ-
Eingaben:++++++++++++++++++
while True:
    print("\n---  Neue  Testiteration:  Geben  Sie  andere  KQ-Werte  ein  oder
brechen Sie das Skript mit STRG+C ab ---")

    try:
        # Neue KQ-Eingabe
        n_value, p1, p2 = module_2_kq_usermodule()

        # Neue kra1-Eingabe
        kra1_input = input("\nBitte geben Sie die KRA1-Raumadressierung ein
(z.B. 111,288,...): ").strip()
        kra1_indices  =  [int(x.strip())  for  x  in  kra1_input.split(",")  if
x.strip().isdigit()]

        # Bytezuordnung erneut berechnen mit neuer KQ-Konfiguration
        reverse_mapping = generate_kra1_byte_mapping(n_value, p1, p2)
        reverse_dict = {v: k for k, v in reverse_mapping.items()}
        reconstructed_bytes  =  [reverse_dict.get(index,  0)  for  index  in
kra1_indices]

        # Ausgabe
        print("\nRekonstruierte Bytewerte:")
        print(reconstructed_bytes)
```

```
        try:
            print("\nRekonstruierter Text aus kra1:")
            print(bytes(reconstructed_bytes).decode("utf-8"))
        except UnicodeDecodeError:
            print("Fehler: Die Bytes konnten nicht korrekt als UTF-8 decodiert werden.")

    except KeyboardInterrupt:
        print("\nSkript wurde vom Benutzer abgebrochen.")
        break
    except Exception as e:
        print(f"Ein Fehler ist aufgetreten: {e}")
```