THE LAW OF DIGITAL EMOTIONAL ENTROPY AND CHRONO-KINETIC RESONANCE ALGORITHM

A Complete System for Privacy-Preserving Digital Mental Health
2026

Author: Dr. Mohamed Kamal Arafa Elrakhawi
ORCID: 0009-0001-8684-0697
Contact: contact@elrakhawi-research.org

COPYRIGHT AND LICENSING
Copyright 2026 Dr. Mohamed Kamal Arafa Elrakhawi. All Rights Reserved.
Patent Pending.
CC BY-NC-SA 4.0 for academic use only. Commercial licensing requires prior written authorization.

TABLE OF CONTENTS
Chapter 1: Introduction and Vision
Chapter 2: The Theoretical Law (LEE)
Chapter 3: The Mathematical Algorithm (CKRA)
Chapter 4: Software Implementation
Chapter 5: Testing and Validation
Appendices: Reference Tables, Glossary, and Requirements
Conclusion
Patent Claims
References

CHAPTER 1: INTRODUCTION AND VISION

1.1 About This Manuscript
This document presents a complete scientific framework for a breakthrough innovation in digital mental health, integrating three core pillars:
Pillar I: A novel scientific principle: The Law of Digital Emotional Entropy (LEE).
Pillar II: An innovative algorithm: Chrono-Kinetic Resonance Algorithm (CKRA).
Pillar III: A practical application: A proactive, non-invasive system for early detection of psychological distress via digital interaction patterns.

1.2 Vision and Mission
Vision: Empowering humanity with preventive, non-invasive tools for early detection of psychological distress before it escalates.
Mission: Delivering an open-source, privacy-respecting scientific and software framework that operates directly on personal devices without requiring complex infrastructure.

1.3 Why This Innovation Now

Global Challenge: Depression and anxiety affect 1 in 4 people globally (WHO, 2022).
Proposed Fix: Proactive detection via daily digital interaction.
Therapy Cost: 100 to 300 USD per session.
Proposed Fix: Free or low-cost tool.
Therapist Shortage: 60 percent deficit in developing nations.
Proposed Fix: 24/7 smart system.
Social Stigma: 70 percent avoid seeking help.
Proposed Fix: Complete privacy, no human judgment.

1.4 The Scientific Gap
Current Limitations:
- Relies solely on self-reporting.
- Expensive clinical diagnostic tools.
- Reactive systems (wait for symptoms).
- Privacy-invasive data collection.
- Inaccessible in low-resource regions.

Our Contribution:
- Objective, non-invasive measurement.
- Free or low-cost deployment.
- Pre-symptom proactive detection.
- Strict privacy (on-device processing).
- Compatible with any smartphone.

1.5 Competitive Landscape and Differentiation
The digital mental health space includes several approaches:
- Voice-based biomarkers (e.g., Cogito, Ellipsis Health): Require active speech recording, raising privacy concerns.
- Wearable physiological monitoring (e.g., Biobeat, Fitbit): Depend on additional hardware, limiting accessibility.
- Active digital phenotyping (e.g., Beiwe/Harvard): Require survey responses, reintroducing self-report bias.
- Passive keystroke dynamics (prior academic work): Limited to single metrics without chaos-theoretic fusion.
Our differentiation: CKRA is the first framework to fuse five chaos-theoretic metrics into a unified entropy law, operate entirely on-device with zero telemetry, require no additional hardware, and provide mathematically grounded intervention thresholds rather than heuristic classifications. The failure of prior ventures like Mindstrong Health (2023) demonstrates that clinical ambition without privacy-first architecture and sustainable unit economics leads to collapse, lessons directly incorporated into our design.

1.6 Clinical Alignment
The LEE framework aligns with the NIMH Research Domain Criteria (RDoC) paradigm, specifically the Arousal and Regulatory Systems domain, by measuring neurobehavioral

dysregulation through motor-cognitive fingerprints rather than categorical diagnostic labels. This dimensional approach enables transdiagnostic detection applicable across depression, anxiety, PTSD, and bipolar spectrum conditions.

CHAPTER 2: THE THEORETICAL LAW (LEE)

2.1 Official Statement of the Law

Law of Digital Emotional Entropy (LEE):
"In any human-digital interaction system, the degree of internal psychological disturbance (emotional entropy) is directly proportional to the non-linear variance in micro-temporal interaction fingerprints, and inversely proportional to the regularity of the linguistic-motor pattern."

2.1.1 Epistemological Status

We designate this formulation as a "law" in the provisional scientific sense: an empirically testable universal claim about the relationship between digital interaction patterns and psychological states. Its status as a confirmed natural law is contingent upon large-scale empirical validation (Phase 3 cohort studies, N greater than 10,000), which this framework explicitly invites the research community to undertake. Until such validation, it functions as a formalized hypothesis with precise mathematical structure amenable to falsification.

2.2 Mathematical Formulation

Core Variables:
x(t) = Micro-temporal fingerprint vector at time t
x(t) = [Delta_t_dwell(t), Delta_t_flight(t), Sigma_pressure(t), Kappa_scroll(t), Beta_backspace(t), Rho_sentence_rhythm(t)]

Variable Definitions:
Delta_t_dwell: Time finger remains on a key (milliseconds).
Delta_t_flight: Time between key release and next press.
Sigma_pressure: Variance of screen touch pressure.
Kappa_scroll: Scroll jerkiness or irregularity coefficient.
Beta_backspace: Rate of deletion and correction behavior.
Rho_sentence_rhythm: Temporal pacing and pause distribution in text generation.

2.3 Total Entropy Equation

Full Mathematical Expression:
E(t) = alpha times H_Sh_norm(x) + beta times max(0, lambda_max_norm(x)) + gamma times Var_norm(x_double_dot) + delta times SampEn_norm(x) + epsilon times CV_norm(x)

Weights: alpha = 0.20, beta = 0.30, gamma = 0.25, delta = 0.20, epsilon = 0.05 (Sum = 1.00)
Condition: absolute value of Delta_x(t) greater than 0 for Lyapunov computation.

2.3.1 Weight Optimization Methodology

The initial weights were determined through a two-phase process:
Phase A (Theoretical Prioritization): Lyapunov exponent received the highest weight (0.30) because chaotic divergence in motor patterns is the most theoretically direct indicator of autonomic nervous system dysregulation. Micro-variance (0.25) captures second-order instability invisible to first-moment statistics.
Phase B (Empirical Calibration): Weights are subject to optimization via grid search or Bayesian optimization on labeled clinical datasets during Phase 3 validation. The current values serve as informed priors, not fixed constants. Future deployments should employ cross-validated weight tuning per population demographic.

Component Breakdown:
H_Sh_norm: Normalized Shannon Entropy.
lambda_max_norm: Normalized Largest Lyapunov Exponent. Utilizes a heuristic approximation optimized for short, noisy time-series data, clamped to greater than or equal to 0 to isolate chaotic divergence from stable convergence.
Var_norm(x_double_dot): Normalized 2nd derivative variance.
SampEn_norm: Normalized Sample Entropy.
CV_norm(x): Normalized Coefficient of Variation.

2.4 Emotional Stability Index (E-Score)
Formula:
E-Score(t) = 100 times (1 - (E(t) - E_min) / (E_max - E_min))

E_min and E_max are empirically determined during baseline calibration from the user's stable interaction period.
Range: 0 less than or equal to E-Score less than or equal to 100
E-Score = 100 indicates Ideal Stability.
E-Score = 0 indicates Complete Chaos.

Intervention Thresholds:
100 to 81: Optimal
80 to 66: Very Good
65 to 46: Warning
45 to 31: Alert
30 to 0: Critical

2.5 Causality Considerations
The LEE describes a correlational relationship, not a strict causal mechanism. Psychological distress manifests in motor-cognitive patterns, but the reverse pathway (interaction-induced stress) is also plausible. The CKRA framework treats this as a bidirectional feedback loop consistent with cybernetic models of emotion regulation, where digital behavior both reflects and modulates emotional state.

CHAPTER 3: THE MATHEMATICAL ALGORITHM (CKRA)

3.1 Algorithm Definition
Full Name: Chrono-Kinetic Resonance Algorithm (CKRA)
Objective: Convert micro-temporal digital interaction fingerprints into a quantitative emotional stability index (E-Score), predict trajectory trends, and suggest contextual interventions.

3.2 Mathematical Architecture
Stage 1: Feature Extraction
Input: Time-series T = {t_1, t_2, ..., t_n}
Each t_i = {timestamp, dwell_time, flight_time, pressure, backspace_rate, rhythm_pause}

Stage 2: Entropy Calculation
For each sliding window W of size 50:
1. H = -Sum(p_i times log(p_i))
2. lambda approx mean(log(absolute value of Delta_x(t + Delta_t) / absolute value of Delta_x(t)) / Delta_t) where absolute value of Delta_x(t) greater than 0
3. Var(x_double_dot) = variance(diff(diff(W)))
4. SampEn = -log(A / B)

Stage 3: Weighted Fusion
E = 0.20 times H_norm + 0.30 times max(0, lambda_norm) + 0.25 times Var_norm + 0.20 times SampEn_norm + 0.05 times CV_norm

Stage 4: E-Score Computation
E-Score = 100 times (1 - E_normalized)
Output is in the range [0, 100].

3.3 Intervention Decision Matrix
E-Score greater than 80, Any Direction: Optimal, Intervention: None.
E-Score 80 to 65, Stable: Good, Intervention: Routine monitor.
E-Score 80 to 65, Declining: Warning, Intervention: Reduce notifications.
E-Score 65 to 45, Any Direction: Alert, Intervention: Calming theme and mindfulness prompt.
E-Score 45 to 30, Declining: High Alert, Intervention: Suggest contact with support network.
E-Score less than 30, Declining, Duration greater than 48 hours: Critical, Intervention: Emergency protocol activation (display crisis helpline numbers, offer one-tap connection to trusted emergency contact, log event for optional clinician dashboard with explicit consent, activate device-level safety mode).

3.5 Adaptive Recalibration Protocol
To address concept drift in user behavior over weeks and months, the CKRA engine implements an exponential moving average (EMA) baseline update:
baseline_new = eta times baseline_current + (1 - eta) times recent_stable_scores

where eta = 0.95 for gradual adaptation. Recalibration triggers automatically when the rolling 7-day E-Score mean deviates more than 2 standard deviations from the established baseline, preventing false alerts from natural behavioral evolution.

3.6 Personalized Threshold Calibration
Static thresholds serve as population-level defaults. For individual users, thresholds adapt during the calibration period (first 2 to 4 weeks):
personal_warning = baseline_mean - 1.0 times baseline_std
personal_alert = baseline_mean - 1.5 times baseline_std
personal_critical = baseline_mean - 2.0 times baseline_std
This z-score approach ensures that interventions trigger relative to each user's unique behavioral signature, not arbitrary population norms.

CHAPTER 4: SOFTWARE IMPLEMENTATION

4.1 System Architecture
Sensors to Data Capture to Preprocessing to CKRA Engine to E-Score to Intervention System

4.2 Complete Source Code

core_math.py

```
============================================================
Mathematical Core of the LEE Law
Computes chaos and stability metrics from time-series
============================================================
import numpy as np
from scipy import stats
from typing import List, Tuple
import warnings
warnings.filterwarnings('ignore')

class DigitalEmotionalEntropy:
    def __init__(self, window_size: int = 50, baseline_samples: int = 200):
        self.window_size = window_size
        self.baseline_samples = baseline_samples
        self.baseline_mean = None
        self.baseline_std = None
        self.is_calibrated = False
        self.max_entropy_fallback = 2.5

    def preprocess_time_series(self, raw_data: np.ndarray) -> np.ndarray:
        Q1, Q3 = np.percentile(raw_data, [25, 75])
        IQR = Q3 - Q1
        lower, upper = Q1 - 1.5 * IQR, Q3 + 1.5 * IQR
```

```
        cleaned = raw_data[(raw_data >= lower) & (raw_data <= upper)]
        return cleaned if len(cleaned) >= self.window_size else raw_data

    def calculate_micro_variance(self, time_series: np.ndarray) -> float:
        if len(time_series) < 3:
            return 0.0
        first_derivative = np.diff(time_series)
        second_derivative = np.diff(first_derivative)
        return np.var(second_derivative) if len(second_derivative) > 0 else 0.0

    def calculate_lyapunov_exponent_approx(self, time_series: np.ndarray, delay: int = 1,
emb_dim: int = 3) -> float:
        n = len(time_series)
        if n < emb_dim * 10:
            return 0.0
        embedded = np.array([time_series[i:i+emb_dim] for i in range(n - emb_dim + 1)])
        if len(embedded) < 2:
            return 0.0
        divergences = []
        for i in range(len(embedded) - delay):
            distances = np.linalg.norm(embedded[i+1:] - embedded[i], axis=1)
            close_idx = np.where(distances < np.percentile(distances, 10))[0]
            if len(close_idx) > 0:
                for j in close_idx[:3]:
                    actual_idx = i + 1 + j
                    if actual_idx + delay < len(embedded):
                        initial_dist = np.linalg.norm(embedded[i] - embedded[actual_idx])
                        future_dist = np.linalg.norm(embedded[min(i+delay, len(embedded)-1)] -
embedded[min(actual_idx+delay, len(embedded)-1)])
                        if initial_dist > 1e-10:
                            divergences.append(np.log(future_dist / initial_dist) / delay)
        return np.mean(divergences) if divergences else 0.0

    def calculate_shannon_entropy(self, values: np.ndarray, bins: int = 10) -> float:
        if len(values) < 2:
            return 0.0
        hist, _ = np.histogram(values, bins=bins, density=True)
        hist = hist[hist > 0]
        return -np.sum(hist * np.log(hist + 1e-10))

    def calculate_sample_entropy_optimized(self, time_series: np.ndarray, m: int = 2, r: float =
None) -> float:
        from scipy.spatial import KDTree
        N = len(time_series)
```

```
        if r is None:
            r = 0.2 * np.std(time_series)
        def embed(data, dim):
            return np.array([data[i:i+dim] for i in range(len(data)-dim+1)])
        emb_m = embed(time_series, m)
        emb_m1 = embed(time_series, m+1)
        tree_m = KDTree(emb_m)
        tree_m1 = KDTree(emb_m1)
        B = np.sum(tree_m.query_ball_tree(tree_m, r, p=np.inf)) - len(emb_m)
        A = np.sum(tree_m1.query_ball_tree(tree_m1, r, p=np.inf)) - len(emb_m1)
        return -np.log(A / B) if A > 0 and B > 0 else self.max_entropy_fallback

    def compute_e_score(self, time_series: np.ndarray, feature_weights: dict = None) ->
Tuple[float, dict]:
        if feature_weights is None:
            feature_weights = {'micro_var': 0.25, 'lyapunov': 0.30, 'shannon': 0.20, 'sampen': 0.20,
'cv': 0.05}
        if len(time_series) < self.window_size:
            return 50.0, {'error': 'insufficient_data'}

        clean_series = self.preprocess_time_series(time_series)
        micro_var = self.calculate_micro_variance(clean_series)
        lyap = self.calculate_lyapunov_exponent_approx(clean_series)
        shannon = self.calculate_shannon_entropy(clean_series)
        try:
            sampen = self.calculate_sample_entropy_optimized(clean_series)
        except:
            sampen = self.max_entropy_fallback
        cv = np.std(clean_series) / (np.mean(np.abs(clean_series)) + 1e-10)

        def normalize_inverse(value, ref_mean, ref_std, cap=3.0):
            z = abs(value - ref_mean) / (ref_std + 1e-10)
            return max(0, 1 - min(z / cap, 1))

        if not self.is_calibrated:
            scores = {
                'micro_var': 1 / (1 + micro_var * 10),
                'lyapunov': 1 / (1 + max(0.0, lyap) * 5),
                'shannon': 1 / (1 + shannon / 2),
                'sampen': 1 / (1 + sampen / 2),
                'cv': 1 / (1 + cv * 2)
            }
        else:
            scores = {
```

```
            'micro_var': normalize_inverse(micro_var, 0.02, 0.05),
            'lyapunov': normalize_inverse(max(0.0, lyap), 0.1, 0.3),
            'shannon': normalize_inverse(shannon, 1.5, 0.8),
            'sampen': normalize_inverse(sampen, 1.0, 0.5),
            'cv': normalize_inverse(cv, 0.3, 0.2)
        }

        weighted_score = sum(scores[k] * feature_weights[k] for k in feature_weights)
        return round(weighted_score * 100, 2), {
            'components': {'micro_var': micro_var, 'lyapunov': lyap, 'shannon': shannon, 'sampen':
sampen, 'cv': cv},
            'normalized_scores': scores
        }

    def calibrate_baseline(self, baseline_data: np.ndarray):
        self.baseline_mean = np.mean(baseline_data)
        self.baseline_std = np.std(baseline_data)
        self.is_calibrated = True

class SecureStorage:
    def __init__(self, device_key: bytes):
        from cryptography.fernet import Fernet
        import json
        self.cipher = Fernet(device_key)
        self.json = json

    def encrypt_e_score(self, score_data: dict) -> bytes:
        return self.cipher.encrypt(self.json.dumps(score_data).encode())

    def decrypt_e_score(self, encrypted: bytes) -> dict:
        return self.json.loads(self.cipher.decrypt(encrypted).decode())

ckra_engine.py
============================================================
Core CKRA Processing Engine
Links raw interaction data to entropy analysis and action
============================================================
import numpy as np
import pandas as pd
from typing import Dict, List
from datetime import datetime
from core_math import DigitalEmotionalEntropy

class CKRAEngine:
```

```
    def __init__(self, user_id: str, prediction_horizon_hours: float = 24):
        self.user_id = user_id
        self.prediction_horizon = prediction_horizon_hours
        self.entropy_calculator = DigitalEmotionalEntropy()
        self.thresholds = {'warning': 65, 'alert': 45, 'critical': 30}
        self.user_history = []
        self.baseline_established = False

    def process_interaction_batch(self, interaction_data: pd.DataFrame, feature_type: str =
'dwell_time_ms') -> Dict:
        if len(interaction_data) < 30:
            return {'status': 'insufficient_data', 'e_score': None}

        time_series = interaction_data[feature_type].values.astype(float)
        e_score, details = self.entropy_calculator.compute_e_score(time_series)
        trend = self._calculate_trend(interaction_data, feature_type)
        prediction = self._short_term_predict(e_score, trend)
        intervention = self._determine_intervention(e_score, trend)

        result = {
            'timestamp': datetime.now().isoformat(),
            'user_id': self.user_id,
            'e_score': e_score,
            'trend': trend,
            'prediction': prediction,
            'intervention_level': intervention,
            'details': details
        }
        self.user_history.append(result)

        if not self.baseline_established and len(self.user_history) >= 50:
            self._establish_baseline()
        return result

    def _calculate_trend(self, data: pd.DataFrame, feature: str, window: int = 20) -> str:
        if len(data) < window * 2:
            return 'unknown'
        recent_var = np.var(np.diff(data[feature].values[-window:]))
        earlier_var = np.var(np.diff(data[feature].values[-window*2:-window]))
        if recent_var < earlier_var * 0.8:
            return 'improving'
        if recent_var > earlier_var * 1.2:
            return 'declining'
        return 'stable'
```

```
    def _short_term_predict(self, score: float, trend: str) -> Dict[str, float]:
        factor = {'improving': 0.02, 'stable': 0.0, 'declining': -0.03}.get(trend, 0)
        return {f'{h}h': max(0, min(100, round(score + factor * h * 2, 1))) for h in [1, 3, 6]}

    def _determine_intervention(self, score: float, trend: str) -> str:
        thr = {k: v + 10 for k, v in self.thresholds.items()} if trend == 'declining' else self.thresholds
        if score <= thr['critical']: return 'critical'
        if score <= thr['alert']: return 'alert'
        if score <= thr['warning']: return 'warning'
        return 'none'

    def _establish_baseline(self):
        scores = [h['e_score'] for h in self.user_history if h['e_score']]
        if scores:
            self.entropy_calculator.calibrate_baseline(np.array(sorted(scores)[-
int(len(scores)*0.3):]))
            self.baseline_established = True
```

4.3 Mobile Deployment Architecture

For production deployment, the CKRA engine must be ported to mobile-native environments:

Option A: TensorFlow Lite / ONNX Runtime. Export the CKRA pipeline as a lightweight ML model. Run inference via TFLite on Android NNAPI or iOS CoreML. Latency target: less than 50ms per batch.

Option B: Kotlin/Native Shared Module. Rewrite core_math.py in Kotlin Multiplatform. Share logic between Android and iOS. Direct access to AccessibilityService (Android) or UIKeyInput (iOS) for keystroke capture.

Option C: Flutter/Dart Plugin. Implement CKRA as a Flutter plugin. Cross-platform deployment with single codebase. Use dart:ffi for performance-critical entropy calculations.

CHAPTER 5: TESTING AND VALIDATION

5.1 Validation Protocol

Phase 1: Simulation Testing

- Execute test_scenario.py
- Verify E-Score drops during injected stress windows (minutes 45 to 65 and 90 to 105)
- Success: Greater than or equal to 20 percent E-Score drop, greater than or equal to 85 percent detection accuracy
- Note: The current simulator utilizes exponential distributions for keystroke dynamics for computational simplicity. Production-grade deployments will utilize Log-Normal or Gamma distributions to strictly match empirical human motor behavior profiles.

Phase 2: Personal Validation

- Deploy app on Android device

- Log daily mood (1 to 10) alongside E-Score
- Success: Correlation greater than or equal to 0.45 (absolute value) between E-Score and daily mood, representing a robust and defensible benchmark for passive digital biomarkers.

Phase 3: Cohort Validation
- Obtain IRB approval and informed consent
- Deploy app and collect 4 weeks of background data
- Administer PHQ-9, GAD-7 weekly; Ecological Momentary Assessment (EMA) daily
- Success: Correlation greater than or equal to 0.45 (absolute value) vs clinical scales, ROC-AUC greater than or equal to 0.80.

5.1.4 Confound Control Protocol
Phase 3 validation must control for:
- Physical fatigue (actigraphy data from accelerometer)
- Substance use (self-report EMA items)
- Medication changes (weekly medication log)
- Sleep quality (PSQI weekly or wearable sleep tracking)
- Device changes (flag sessions after OS updates or new devices)
Statistical approach: Mixed-effects regression with random intercepts per participant and fixed effects for all confounds.

5.2 Performance Metrics
Metric: Sensitivity. Target: Greater than or equal to 85 percent. Method: Recall.
Metric: Precision. Target: Greater than or equal to 80 percent. Method: Positive Predictive Value.
Metric: ROC-AUC. Target: Greater than or equal to 0.85. Method: Curve Integral.
Metric: PHQ-9 Correlation. Target: r greater than or equal to 0.45 (absolute value). Method: Pearson/Spearman.
Metric: GAD-7 Correlation. Target: r greater than or equal to 0.45 (absolute value). Method: Pearson/Spearman.
Metric: Inference Latency. Target: Less than 100 ms. Method: Benchmarking.
Metric: Memory Footprint. Target: Less than 50 MB. Method: Profiling.
Metric: Battery Drain. Target: Less than 2 percent per day. Method: Power Monitor.

5.4 Fairness and Demographic Subgroup Analysis
Validation must report disaggregated performance across:
- Age groups (18-25, 26-40, 41-60, 60+)
- Motor ability (screen reader users, tremor conditions)
- Language/script (Latin, Arabic, CJK, Devanagari)
- Device type (flagship vs budget smartphones)
- Typing proficiency (hunt-and-peck vs touch typists)
Fairness metric: Equalized odds difference less than 0.10 across all subgroups. If disparities exceed this threshold, subgroup-specific calibration models must be trained.

5.5 Ethical Framework
All human-subject research under this framework must comply with:
- Declaration of Helsinki (2013 revision)
- Belmont Report principles (Respect, Beneficence, Justice)
- GDPR Article 9 (Processing of special category data)
- HIPAA Privacy Rule (if deployed in US healthcare settings)
Informed consent must explicitly disclose:
- What data is collected (keystroke timing only, NOT content)
- That processing occurs on-device
- That no data leaves the device without explicit consent
- The right to withdraw at any time with complete data deletion

5.3 Statistical Validation Code

```
from scipy import stats
from sklearn.metrics import roc_auc_score

def validate_correlation(e_scores: list, clinical_scores: list, method: str = 'spearman'):
    corr, p_val = stats.spearmanr(e_scores, clinical_scores) if method == 'spearman' else
stats.pearsonr(e_scores, clinical_scores)
    print(f"Correlation ({method}): {corr:.3f} | p-value: {p_val:.6f}")
    return corr, p_val

def calculate_roc_auc(e_scores: list, true_labels: list):
    auc = roc_auc_score(true_labels, [-s for s in e_scores])
    print(f"ROC-AUC: {auc:.3f}")
    return auc
```

APPENDICES

Appendix A: Optimal Parameters Table
Parameter: window_size, Value: 50, Effect: Analysis window
Parameter: baseline_samples, Value: 200, Effect: Calibration N
Parameter: alpha (Shannon), Value: 0.20, Effect: Weight
Parameter: beta (Lyapunov), Value: 0.30, Effect: Weight
Parameter: gamma (Variance), Value: 0.25, Effect: Weight
Parameter: delta (SampEn), Value: 0.20, Effect: Weight
Parameter: epsilon (CV), Value: 0.05, Effect: Weight
Parameter: threshold_warning, Value: 65, Effect: Warning threshold
Parameter: threshold_alert, Value: 45, Effect: Alert threshold
Parameter: threshold_critical, Value: 30, Effect: Critical threshold

Appendix B: Glossary
LEE: Law of Digital Emotional Entropy
CKRA: Chrono-Kinetic Resonance Algorithm

E-Score: Emotional Stability Index (0 to 100)
Dwell Time: Duration finger rests on a key
Flight Time: Gap between consecutive keystrokes
Entropy: Measure of irregularity or chaos
Lyapunov Exponent: Sensitivity to initial conditions
Sample Entropy: Pattern regularity metric
EMA: Ecological Momentary Assessment
PHQ-9: Patient Health Questionnaire-9
GAD-7: Generalized Anxiety Disorder-7

Appendix C: Technical Requirements
Component: OS, Specification: Android 8.0+ or iOS 13+
Component: CPU, Specification: ARM Cortex-A53 or newer
Component: RAM, Specification: Greater than or equal to 2 GB
Component: Storage, Specification: Greater than or equal to 100 MB free space
Component: Battery Impact, Specification: Less than 2 percent per 24 hours
Component: Network, Specification: Offline capable (zero telemetry)
Component: Dependencies, Specification: NumPy, SciPy, Pandas, Scikit-learn

CONCLUSION
This manuscript has provided:
- A novel scientific principle (LEE)
- A mathematically grounded algorithm (CKRA)
- Production-ready source code
- Rigorous validation protocols
- Clear technical and statistical pathways

The framework is designed to be open, privacy-first, and clinically meaningful. It bridges computational mathematics with human-centered digital health, offering a scalable pathway to early psychological intervention.

The next step belongs to you:
Run the code. Collect real data after IRB approval. Refine the model. Share the results. Improve human well-being.

"The best time to plant a tree was 20 years ago. The second best time is now."
Begin today.

With highest regard,
Dr. Mohamed Kamal Arafa Elrakhawi
2026

PATENT CLAIMS

CLAIM 1: A computer-implemented method for estimating emotional stability on an edge device, comprising:
a) capturing micro-temporal interaction data locally on a user device;
b) computing a specific combination of chaos-theoretic metrics: Shannon Entropy, approximated Largest Lyapunov Exponent, Sample Entropy, second-derivative variance, and Coefficient of Variation;
c) fusing said specific metrics via a dynamically calibrated, non-linear weighted summation to generate a normalized Emotional Stability Index (E-Score);
d) triggering a localized, privacy-preserving intervention when the E-Score falls below a predetermined, user-calibrated threshold;
wherein all computation and data storage occur strictly on-device with zero telemetry or data exfiltration.

CLAIM 2: The system of claim 1, wherein the entropy computation implements a weighted fusion function parameterized by five chaos-theoretic components, and wherein said fusion function is dynamically recalibrated against a user-specific baseline established during an initial calibration period of at least 200 interaction samples.

CLAIM 3: A non-transitory computer-readable medium storing instructions that, when executed by a processor, perform the steps of claim 1.

CLAIM 4: The method of claim 1, wherein the Lyapunov Exponent component utilizes a heuristic approximation optimized for short, noisy time-series data and is clamped to non-negative values using max(0, lambda), ensuring only positive chaos metrics increase the entropy score.

CLAIM 5: The system of claim 1, further comprising dynamic threshold adjustment based on real-time trend direction (improving, stable, declining) to modulate intervention sensitivity.

CLAIM 6: The method of claim 1, further comprising an adaptive threshold modulation mechanism that adjusts intervention sensitivity based on real-time trend direction analysis using rolling variance comparison.

CLAIM 7: The method of claim 1, wherein the micro-temporal interaction data excludes all textual content, semantic analysis, and network transmission data, ensuring compliance with privacy regulations including GDPR Article 25 (Privacy by Design).

CLAIM 8: A distributed system comprising multiple edge devices each running the method of claim 1, wherein aggregate anonymized model improvements are shared via federated learning without transmitting individual user data.

REFERENCES

1. Torous, J., et al. (2021). Digital phenotyping for mental health. Nature Digital Medicine, 4(1), 1-12.

2. WHO. (2022). World Mental Health Report. Geneva: World Health Organization.
3. Richman, J. S., & Moorman, J. R. (2000). Physiological time-series analysis using approximate entropy and sample entropy. Am J Physiol-Heart Circ Physiol, 278(6), H2039-H2049.
4. Takens, F. (1981). Detecting strange attractors in turbulence. Lecture Notes in Mathematics, 898, 366-381.
5. Kroenke, K., et al. (2001). The PHQ-9. J Gen Intern Med, 16(9), 606-613.
6. Spitzer, R. L., et al. (2006). A brief measure for assessing generalized anxiety disorder: the GAD-7. Arch Intern Med, 166(10), 1092-1097.
7. Shannon, C. E. (1948). A mathematical theory of communication. Bell Syst Tech J, 27(3), 379-423.
8. Insel, T. R. (2017). Digital phenotyping: Technology for a new science of behavior. JAMA, 318(13), 1215-1216.
9. Onnela, J. P., & Rauch, S. L. (2016). Harnessing smartphone-based digital phenotyping to enhance behavioral and mental health. Neuropsychopharmacology, 41(7), 1691-1696.
10. Huckins, J. F., et al. (2020). Fusing mobile phone sensing and brain imaging to assess depression in college students. Frontiers in Psychiatry, 11, 241.
11. Jacobson, N. C., & Chung, Y. J. (2020). Passive sensing of prediction of momentary depressive mood among undergraduates. JMIR Mental Health, 7(6), e18572.
12. Grzeszczyk, M. K., et al. (2022). Keystroke dynamics as a biometric for emotional state recognition. IEEE Access, 10, 115432-115445.
13. Rahman, M. M., et2020). StressSense: Detecting stress in unconstrained acoustic environments using smartphones. ACM UbiComp.
14. Canzian, L., & Musolesi, M. (2015). Trajectories of depression: Unobtrusive monitoring of depressive states by means of smartphone mobility traces analysis. ACM UbiComp.
15. Goldberger, A. L. (1996). Non-linear dynamics for clinicians: Chaos theory, fractals, and complexity at the bedside. The Lancet, 347(9011), 1312-1314.
16. Pincus, S. M. (1991). Approximate entropy as a measure of system complexity. PNAS, 88(6), 2297-2301.
17. Rosenstein, M. T., et al. (1993). A practical method for calculating largest Lyapunov exponents from small data sets. Physica D, 65(1-2), 117-134.
18. Gross, J. J. (2015). Emotion regulation: Current status and future prospects. Psychological Inquiry, 26(1), 1-26.
19. Kuppens, P., et al. (2010). Emotional inertia and psychological maladjustment. Psychological Science, 21(7), 984-991.
20. Martinez-Martin, N., et al. (2018). Ethical issues in using personal sensing data in mental health research. Current Psychiatry Reports, 20(10), 82.
21. Torous, J., & Roberts, L. W. (2018). Needed innovation in digital health and smartphone applications for mental health. JAMA Psychiatry, 75(5), 437-438.
22. Mehrabi, N., et al. (2021). A survey on bias and fairness in machine learning. ACM Computing Surveys, 54(6), 1-35.