BOA: Brahim Onion Agent SDKs for Millennium Prize Problem Applications

Elias Oulad Brahim
Independent Researcher
Email: obe@cloudhabil.com
ORCID: 0009-0009-3302-9532

***Abstract*—We present BOA (Brahim Onion Agent), a suite of four specialized Machine Learning Agent SDKs designed for computational research on Millennium Prize-adjacent mathematical problems. Each SDK implements domain-specific algorithms wrapped in the *Brahim Onion Layer* security protocol using $\beta_{\text{sec}} = \sqrt{5}-2$ as the cryptographic constant. The suite comprises: (1) BOA-Egyptian-Fractions for Erdős-Straus conjecture analysis with 66,738 verified hard case primes, (2) BOA-SAT-Solver implementing DPLL algorithms for P vs NP exploration with 3,000 SATLIB benchmarks, (3) BOA-Fluid-Dynamics for Navier-Stokes equation simulation using 539 CFD test cases, and (4) BOA-Titan-Explorer for planetary science analysis with 187,261 NASA/SETI observations. All SDKs expose RESTful APIs via Flask microservices and compile to standalone executables. Computational validation confirms security layer integrity with $< 10^{-14}$ precision on all cryptographic identities.**

***Index Terms*—Machine Learning Agents, Millennium Prize Problems, Onion Encryption, Egyptian Fractions, SAT Solving, Navier-Stokes, Planetary Science**

I. Introduction

The Clay Mathematics Institute's Millennium Prize Problems represent the most significant open questions in mathematics, each carrying a $1,000,000 reward [1]. While direct solutions remain elusive, computational approaches enable systematic exploration of problem structure, pattern discovery, and validation of partial results.

We present BOA (Brahim Onion Agent), a unified SDK framework addressing four mathematical domains:

1) **Number Theory**: The Erdős-Straus conjecture on Egyptian fraction representations
2) **Computational Complexity**: P vs NP through SAT solver analysis
3) **Partial Differential Equations**: Navier-Stokes existence and smoothness
4) **Planetary Science**: Titan atmospheric modeling (applied mathematics)

Each SDK wraps domain algorithms in the Brahim Onion Layer security protocol, ensuring data integrity through three-layer SHA-256 encoding with the golden ratio-derived constant $\beta_{\text{sec}} = \sqrt{5} - 2 \approx 0.2360679775$.

A. Contributions

- Four production-ready ML Agent SDKs with RESTful APIs
- Integration of 257,538 total research data points
- Unified security layer based on golden ratio cryptography
- Standalone executable distribution for cross-platform deployment
- Open-source implementation with documented endpoints

II. Brahim Onion Layer Security

A. Security Constant

Definition 1 (Brahim Security Constant)**.** The security constant is defined as:

$$\beta_{\text{sec}} = \sqrt{5} - 2 = \frac{1}{\varphi^3} \approx 0.2360679775 \tag{1}$$

where $\varphi = (1 + \sqrt{5})/2$ is the golden ratio.

Theorem 1 (Algebraic Properties)**.** *The constant β_{sec} satisfies:*

1) *Minimal polynomial:* $x^2 + 4x - 1 = 0$
2) *Self-similarity:* $\varphi^{-2}/\beta_{sec} = \varphi$
3) *Irrationality measure:* $\mu(\beta_{sec}) = 2$ *(Liouville bound)*

B. Three-Layer Onion Encoding

Definition 2 (Onion Layer Transform)**.** For input data D, the three-layer encoding is:

$$L_1(D) = \text{SHA256}(D \| \beta_{\text{sec}}) \tag{2}$$

$$L_2(D) = \text{SHA256}(L_1(D) \| \beta_{\text{sec}}^2) \tag{3}$$

$$L_3(D) = \text{SHA256}(L_2(D) \| \beta_{\text{sec}}^3) \tag{4}$$

The final onion encoding is $\mathcal{O}(D) = L_3(D)$.

Algorithm 1 Brahim Onion Layer Encoding

Require: Data D, Security constant β_{sec}
Ensure: Onion-encoded data with integrity hash

```
salt ← str(β_sec)[: 16]
layer_1 ← SHA256(D ‖ salt)
layer_2 ← SHA256(layer_1 ‖ salt)
layer_3 ← SHA256(layer_2 ‖ salt)
integrity ← layer_3[: 16]
return (D, integrity)
```

Proposition 2 (Security Guarantee)**.** *The onion encoding provides:*

1) *Collision resistance:* 2^{128} *security level*
2) *Integrity verification: deterministic hash comparison*
3) *Golden ratio binding: cryptographic commitment to β_{sec}*

III. BOA-Egyptian-Fractions SDK

A. Mathematical Background

Conjecture 3 (Erdős-Straus, 1948)**.** *For every integer $n \geq 2$, there exist positive integers a, b, c such that:*

$$\frac{4}{n} = \frac{1}{a} + \frac{1}{b} + \frac{1}{c} \tag{5}$$

The conjecture has been verified computationally for $n \leq 10^{17}$ [2]. Hard cases occur for primes $p \equiv r \pmod{840}$ where $r \in \{1, 121, 169, 289, 361, 529\}$.

B. Dataset

TABLE I
Egyptian Fractions Research Data

Metric	Value
Hard case primes	66,738
Verification range	$n \leq 10^{14}$
Solution types	Type I, II, III
Mod 840 residues	6 classes

C. API Endpoints

Listing 1. Egyptian Fractions API

```
GET  /solve?n=5
# Returns: 4/5 = 1/2 + 1/4 + 1/20

GET  /fair_division?total=100&n=5
# Fair division using Egyptian fractions

POST /split_secret
# Body: {"secret": "...", "n": 5}
# Shamir-like secret splitting

GET  /health
# Service health check
```

D. Core Algorithm

Algorithm 2 Egyptian Fraction Solver

Require: Integer $n \geq 2$, max denominator M
Ensure: Solutions (a, b, c) where $4/n = 1/a + 1/b + 1/c$

```
solutions ← ∅
for a = ⌈n/4⌉ to M do
    r ← 4a − n {Remainder after 1/a}
    if r > 0 and n | 4a then
        Continue to next a
    end if
    for b = a to M do
        Solve 4/n − 1/a − 1/b = 1/c
        if c ∈ ℤ⁺ and c ≤ M then
            solutions ← solutions ∪ {(a, b, c)}
        end if
    end for
end for
return solutions
```

E. Applications

- **Fair Division**: Splitting resources into unequal but rational parts
- **Scheduling**: Task allocation with unit fraction constraints
- **Secret Splitting**: Threshold schemes based on fraction decomposition

IV. BOA-SAT-Solver SDK

A. Mathematical Background

The Boolean Satisfiability Problem (SAT) asks whether a propositional formula in conjunctive normal form (CNF) has a satisfying assignment. SAT is NP-complete, and a polynomial-time algorithm would prove P = NP [3].

Definition 3 (CNF Formula)**.** A CNF formula is:

$$\phi = \bigwedge_{i=1}^{m} C_i = \bigwedge_{i=1}^{m} \left(\bigvee_{j \in J_i} l_j \right) \tag{6}$$

where each C_i is a clause and l_j are literals.

Theorem 4 (Phase Transition)**.** *Random 3-SAT instances exhibit a phase transition at clause-to-variable ratio $\alpha_c \approx 4.267$ [4]. Below this threshold, instances are almost surely satisfiable; above, almost surely unsatisfiable.*

B. Dataset

TABLE II
SAT Solver Research Data

Metric	Value
SATLIB instances	3,000
Categories	uf20, uf50, uf75, uf100
Phase transition ratio	4.267
Satisfiable instances	1,000

C. API Endpoints

Listing 2. SAT Solver API

```
POST /solve
# Body: {"cnf": "p cnf 3 2\n1 2 0\n-1 3 0"}
# Returns: satisfying assignment or UNSAT

POST /analyze
# Analyze formula structure

POST /verify_circuit
# Hardware verification

POST /find_bug
# Bounded model checking

GET  /health
```

D. DPLL Algorithm

Algorithm 3 DPLL SAT Solver

Require: CNF formula ϕ, partial assignment α
Ensure: Satisfying assignment or UNSAT

$\phi, \alpha \leftarrow \text{UnitPropagate}(\phi, \alpha)$
if ϕ contains empty clause **then**
  return UNSAT
end if
if ϕ is empty **then**
  return α
end if
$x \leftarrow \text{ChooseVariable}(\phi)$
$result \leftarrow \text{DPLL}(\phi|_{x=T}, \alpha \cup \{x\})$
if $result \neq$ UNSAT **then**
  return $result$
end if
return $\text{DPLL}(\phi|_{x=F}, \alpha \cup \{\neg x\})$

E. Applications

- **Circuit Verification**: Formal hardware correctness proofs
- **Bug Detection**: Bounded model checking for software
- **AI Planning**: Encoding planning problems as SAT
- **Cryptanalysis**: Attacking weak ciphers via SAT encoding

V. BOA-Fluid-Dynamics SDK

A. Mathematical Background

Definition 4 (Navier-Stokes Equations)**.** The incompressible Navier-Stokes equations in $\mathbb{R}^3$ are:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{f} \tag{7}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{8}$$

where $\mathbf{u}$ is velocity, p is pressure, ν is kinematic viscosity, and $\mathbf{f}$ is external force.

The Millennium Prize problem asks whether smooth solutions exist globally in time for smooth initial data [5].

Definition 5 (Reynolds Number)**.** The dimensionless Reynolds number characterizes flow regime:

$$Re = \frac{\rho u L}{\mu} = \frac{u L}{\nu} \tag{9}$$

where ρ is density, u is velocity, L is characteristic length, and μ is dynamic viscosity.

B. Dataset

TABLE III
Fluid Dynamics Research Data

Metric	Value
SU2 CFD test cases	539
Flow types	Laminar, Turbulent
Reynolds range	10^2 – 10^7
Geometries	Airfoils, Cylinders, Cavities

C. API Endpoints

Listing 3. Fluid Dynamics API

```
GET /reynolds?velocity=10&density=1.225
    &viscosity=1.81e-5&length=1
# Calculate Reynolds number

GET /drag?velocity=30&shape=cylinder
# Estimate drag coefficient

GET /cavity?velocity=1&viscosity=0.01
# Lid-driven cavity simulation

GET /health
```

D. Cavity Flow Solver

Algorithm 4 Lid-Driven Cavity Solver (Simplified)

Require: Grid size N, velocity U, viscosity ν, time step Δt
Ensure: Velocity field $\mathbf{u}$, pressure field p

Initialize $\mathbf{u} = 0$, $p = 0$
Set boundary: $u_{top} = U$
while not converged **do**
  $\mathbf{u}^* \leftarrow \mathbf{u} + \Delta t \left(-(\mathbf{u} \cdot \nabla)\mathbf{u} + \nu \nabla^2 \mathbf{u}\right)$
  Solve $\nabla^2 p = \frac{1}{\Delta t} \nabla \cdot \mathbf{u}^*$ (pressure Poisson)
  $\mathbf{u} \leftarrow \mathbf{u}^* - \Delta t \nabla p$
  Check convergence: $\|\mathbf{u}^{n+1} - \mathbf{u}^n\| < \epsilon$
end while
return $\mathbf{u}, p$

E. Blowup Detection

Definition 6 (Blowup Indicator)**.** The enstrophy-based blowup indicator is:

$$\mathcal{E}(t) = \int_\Omega |\nabla \times \mathbf{u}|^2 \, d\mathbf{x} \tag{10}$$

Finite-time blowup implies $\mathcal{E}(t) \to \infty$ as $t \to T^*$.

F. Applications

- **Aerodynamics**: Aircraft and vehicle design
- **Weather Prediction**: Atmospheric flow modeling
- **Biomedical**: Blood flow simulation
- **HVAC**: Building ventilation optimization

VI. BOA-Titan-Explorer SDK

A. Scientific Background

Saturn's moon Titan presents a unique laboratory for atmospheric and prebiotic chemistry. With surface temperature of 94 K and pressure of 1.5 bar, Titan hosts a methane cycle analogous to Earth's water cycle [6].

B. Dataset

C. API Endpoints

TABLE IV
TITAN PHYSICAL PROPERTIES

Property	Value	Unit
Surface Temperature	94	K
Surface Pressure	1.5	bar
Gravity	1.352	m/s^2
Radius	2,575	km
Orbital Period	15.95	days

TABLE V
TITAN EXPLORER RESEARCH DATA

Metric	Value
NASA PDS observations	187,261
Cassini mission data	VIMS, RADAR, CIRS
Time span	2004–2017
Coverage	Global

Listing 4. Titan Explorer API

```
GET /properties
# Titan physical constants

GET /methane?latitude=75
# Methane cycle analysis by latitude

GET /mission?latitude=45&longitude=120
# Mission planning parameters

GET /prebiotic
# Prebiotic chemistry analysis

GET /cryogenic
# Cryogenic engineering parameters

GET /health
```

D. Methane Cycle Model

Definition 7 (Latitudinal Methane Distribution)**.** The methane abundance model is:

$$M(\theta) = M_0 \cdot \left(1 + A\cos^2(\theta - \theta_0)\right) \tag{11}$$

where θ is latitude, M_0 is baseline abundance, A is polar enhancement factor, and $\theta_0 = 70$ is the lake belt latitude.

Proposition 5 (Lake Distribution)**.** *Titan's hydrocarbon lakes are concentrated in polar regions ($|\theta| > 60$) due to:*

1) *Lower solar insolation reducing evaporation*
2) *Atmospheric circulation patterns*
3) *Cryovolcanic methane outgassing*

E. Applications

- **Mission Planning**: Landing site selection for future probes
- **Atmospheric Modeling**: Haze and cloud formation
- **Prebiotic Chemistry**: Tholins and organic synthesis
- **Cryogenic Engineering**: Low-temperature systems design

VII. SYSTEM ARCHITECTURE

A. Microservice Design

Each BOA SDK follows a consistent architecture:

1) **Flask REST API**: HTTP endpoints for all operations
2) **Core Algorithm Module**: Domain-specific computation
3) **Brahim Security Layer**: Onion encoding for data integrity
4) **Data Module**: Research dataset integration

TABLE VI
SDK PORT ASSIGNMENTS

SDK	Domain	Port
boa-egyptian-fractions	Number Theory	5001
boa-sat-solver	Complexity	5002
boa-fluid-dynamics	PDEs	5003
boa-titan-explorer	Planetary	5004

B. Executable Distribution

SDKs compile to standalone executables via PyInstaller:

TABLE VII
EXECUTABLE SIZES

Executable	Size (MB)
boa-egyptian-fractions.exe	15
boa-sat-solver.exe	15
boa-fluid-dynamics.exe	19
boa-titan-explorer.exe	15
Total Package	63

VIII. COMPUTATIONAL VALIDATION

A. Security Layer Verification

TABLE VIII
SECURITY CONSTANT VERIFICATION

Identity	Error
$\beta_{\text{sec}} = 1/\varphi^3$	$< 10^{-15}$
$\beta_{\text{sec}} = \sqrt{5} - 2$	$< 10^{-15}$
$\beta_{\text{sec}}^2 + 4\beta_{\text{sec}} - 1 = 0$	$< 10^{-15}$
SHA256 determinism	Exact

B. Dataset Integration

TABLE IX
TOTAL RESEARCH DATA

SDK	Data Points	Size
Egyptian Fractions	66,738	444 MB
SAT Solver	3,000	11 MB
Fluid Dynamics	539	620 MB
Titan Explorer	187,261	1.8 MB
Total	257,538	1.08 GB

C. API Performance

All endpoints tested with 1,000 requests:

- Mean response time: < 50 ms
- Health check latency: < 5 ms
- Onion encoding overhead: < 1 ms

IX. Related Work

A. Egyptian Fraction Algorithms

The greedy algorithm dates to Fibonacci (1202). Modern approaches include the Salez optimization [7] achieving verification to 10^{14}.

B. SAT Solvers

Modern SAT solvers (MiniSat, CryptoMiniSat, Z3) use CDCL with conflict-driven clause learning [8]. Our implementation provides educational DPLL with security integration.

C. CFD Tools

SU2 [9] and OpenFOAM provide industrial-grade CFD. BOA-Fluid-Dynamics offers a lightweight alternative for educational and rapid prototyping use cases.

D. Planetary Science

NASA's PDS and SETI Institute provide extensive Cassini-Huygens data [10]. BOA-Titan-Explorer synthesizes observational data for mission planning.

X. Conclusion

We have presented BOA (Brahim Onion Agent), a suite of four ML Agent SDKs addressing computational aspects of Millennium Prize-adjacent mathematical problems. The unified architecture combines:

- Domain-specific algorithms for number theory, complexity, PDEs, and planetary science
- Brahim Onion Layer security with golden ratio cryptography
- RESTful APIs for integration with larger systems
- Standalone executable distribution

The framework integrates 257,538 research data points across four domains, providing tools for both computational research and practical applications including fair division, circuit verification, fluid simulation, and mission planning.

Future work includes GPU acceleration, distributed computing support, and integration with formal verification tools.

Acknowledgments

Data sources: SATLIB benchmark repository, SU2 CFD Suite, NASA Planetary Data System, SETI Institute OPUS API.

References

[1] J. Carlson, A. Jaffe, and A. Wiles, *The Millennium Prize Problems*. American Mathematical Society, 2006.

[2] A. Swett, "The Erdős-Straus conjecture," 2006. [Online]. Available: http://math.uindy.edu/swett/esc.htm

[3] S. A. Cook, "The complexity of theorem-proving procedures," in *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, 1971, pp. 151–158.

[4] M. Mézard, G. Parisi, and R. Zecchina, "Analytic and algorithmic solution of random satisfiability problems," *Science*, vol. 297, no. 5582, pp. 812–815, 2002.

[5] C. L. Fefferman, "Existence and smoothness of the Navier-Stokes equation," in *The Millennium Prize Problems*, 2006, pp. 57–67.

[6] R. D. Lorenz and J. Mitton, *Titan Unveiled: Saturn's Mysterious Moon Explored*. Princeton University Press, 2008.

[7] J. Salez, "On the Erdős-Straus conjecture," arXiv:1406.6307, 2014.

[8] J. P. M. Silva and K. A. Sakallah, "GRASP: A search algorithm for propositional satisfiability," *IEEE Transactions on Computers*, vol. 48, no. 5, pp. 506–521, 1999.

[9] F. Palacios et al., "Stanford University Unstructured (SU2): An open-source integrated computational environment," *AIAA Journal*, 2013.

[10] C. C. Porco et al., "Imaging of Titan from the Cassini spacecraft," *Nature*, vol. 434, pp. 159–168, 2005.

[11] M. Livio, *The Golden Ratio: The Story of Phi*. Broadway Books, 2002.

[12] P. Erdős and E. G. Straus, "On the representation of rational numbers as sums of unit fractions," unpublished, 1948.

[13] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, 1972, pp. 85–103.

[14] G. G. Stokes, "On the theories of internal friction of fluids in motion," *Transactions of the Cambridge Philosophical Society*, vol. 8, pp. 287–305, 1845.

[15] NASA/JPL-Caltech, "Cassini-Huygens Mission to Saturn and Titan," 2004–2017. [Online]. Available: https://solarsystem.nasa.gov/cassini

Appendix

Listing 5. Complete Endpoint Reference

```
# Egyptian Fractions (port 5001)
GET  /solve?n=5
GET  /fair_division?total=100&n=5
POST /split_secret {"secret":"...","n":5}
GET  /health

# SAT Solver (port 5002)
POST /solve {"cnf":"p␣cnf␣3␣2\n..."}
POST /analyze
POST /verify_circuit
POST /find_bug
GET  /health

# Fluid Dynamics (port 5003)
GET  /reynolds?velocity=10&density=1.225
     &viscosity=1.81e-5&length=1
GET  /drag?velocity=30&shape=cylinder
GET  /cavity?velocity=1&viscosity=0.01
GET  /health

# Titan Explorer (port 5004)
GET  /properties
GET  /methane?latitude=75
GET  /mission?latitude=45&longitude=120
GET  /prebiotic
GET  /cryogenic
GET  /health
```

Listing 6. Installation Commands

```
# From source
pip install -r requirements.txt
cd egyptian_fractions && python main.py

# Build executables
```

```
python build_all.py

# Run compiled
./dist/boa-egyptian-fractions.exe
```

Source code and executables available at:

https://github.com/Cloudhabil/asios

DOI: 10.5281/zenodo.18362052