Layer 1 Spec-Sheet: Hybrid Classical–Quantum Chaos Entropy Generator

Purpose

Generate a cryptographically strong 256-bit master key by combining entropy from four independent sources:

- **Classical chaos**: Multi-particle nonlinear dynamics of 300 particles in a pulsing elliptical container.
- **Quantum-inspired chaos**: Kicked-rotor unitary evolution and von Neumann entropy extraction.
- **Hardware RNG**: `os.urandom()`, system timestamp, process ID.
- **Post-quantum lattice mixing + KDF**: Polynomial mixing in $\mathbb{Z}_q[X]/(X^n + 1)$ and Argon2id or PBKDF2-SHA3-512.

This layered design ensures:

- Independence of all entropy sources
- Resistance to classical and quantum adversaries
- Forward secrecy via fresh salt and hardware randomness per run

Inputs

Name	Value	Description
NUM_BALLS	300	Number of particles in classical sim
SIM_FRAMES	1500	Number of simulation frames
BALL_DIM	256	Dimension of quantum state matrix
PBKDF2_ITERS	300 000	PBKDF2 iterations (if Argon2 unavailable)
QUANTUM_ITERATIONS	600	Quantum chaos evolution steps
LATTICE_DIM	512	Polynomial degree for lattice mixer
LATTICE_MODULUS	3329	Prime modulus (Kyber-compatible)
ARGON2_TIME	3	Argon2id time cost
ARGON2_MEMORY	32 MiB	Argon2id memory cost
ARGON2_PARALLELISM	CPU cores	Argon2id parallelism parameter

Additional entropy: hardware RNG, high-resolution timestamp, PID.

Outputs

- **Master key**: 32-byte (256-bit) SHA3-256 digest output (hex-encoded)
- **Metadata**: classical/quantum entropy bits, security level, entropy sources list, performance metrics

Component Breakdown

1. Classical Chaos Simulation

- **Elliptical container** with semi-axes
$$a(t) = a_0 + A\sin(\omega t), \quad b(t) = b_0 + A\cos(\omega t)$$
- **Chaotic map** per particle
$$x_{n+1} = r\,x_n(1 - x_n) + \epsilon, \quad \epsilon \sim U(-\delta, \delta)$$
- **Force calculation** (Newtonian + noise + spin coupling)
$$F_{ij} = G\frac{m_i m_j}{d_{ij}^2}, \quad v \leftarrow v + \frac{F}{m}\Delta t$$
- **Collision handling**: elastic reflection
$$v_i' = v_i - 2\frac{(v_i - v_j)\cdot n}{1 + m_j/m_i}\,n$$
- **Entropy capture**: Stream SHA3-512 over state variables $(x, y, v_x, v_y, \ldots)$

2. Quantum Chaos Engine

- **Kicked rotor Hamiltonian**
$$H(t) = \tfrac{p^2}{2} + K\cos(\theta)\sum_n \delta(t - nT)$$
- **Unitary evolution**
$$U = e^{-iH\Delta t/\hbar}, \quad \psi_{t+1} = U\psi_t$$
- **Entropy extraction**: compute von Neumann entropy
$$S = -\mathrm{Tr}(\rho \ln \rho),\ \ \rho = |\psi\rangle\langle\psi|$$
and stream SHA3-256 of entropy and phase data.

3. Lattice Mixing

- **Polynomial ring**: $R_q = \mathbb{Z}_q[X]/(X^n + 1)$, with $n = 512$, $q = 3329$.
- **Encoding**: map SHA3-512 output bytes to normalized floats then to polynomial coefficients via $\lfloor b\,q/256 \rfloor$.
- **Mixing**:
  - **NTT (fast)**: requires Kyber `zetas[]` and `inv_zetas[]` for Cooley–Tukey NTT.
  - **Convolution (fallback)**: fold convolution result by $X^n = -1$.

4. Key Derivation Function

1. **Combine** domain-separated hashes:

   ```
   SHA3-512(0x01||classical_entropy) ||
   SHA3-512(0x02||quantum_entropy) ||
   SHA3-512(0x03||lattice_entropy) ||
   SHA3-512(0x04||hardware_entropy)
   ```

2. **Argon2id** (t=3, mem=32 MiB, paral=cores) or **PBKDF2-SHA3-512** (300 000 iterations)
3. **Strengthen**: SHA3-512(stage1_key||salt)
4. **Lattice mix**: feed stage2 output through lattice mixer
5. **Finalize**: SHA3-256(stage2||lattice_mix||salt)

Security & Performance

- **Post-quantum security**: relies on MLWE hardness and SHA3-512
- **Entropy sources**: ≥4 independent
- **Forward secrecy**: fresh salt & hardware RNG per invocation
- **Performance**:
  - Classical sim: ~50 s (@30 FPS)
  - Quantum evolution: ~10 s
  - Lattice mixing: <1 s (NTT) or ~2 s (convolution)
  - KDF: ~0.5 s (Argon2id) or ~0.2 s (PBKDF2)

Integration Notes

- All parameters are configurable in `layer1.py`
- Toggle `USE_KYBER_NTT` and supply Kyber zetas to enable fast mixing
- Components (classical, quantum, lattice, KDF) are modular and testable

End of Spec-Sheet