Quantization Techniques in Chaos-Based Pseudo-Random Numbers Generators

Kafui Homevo

February 2026

1 Introduction

Chaotic systems, because of their impredictability, had in the 19s fascinated the industry. And soon, their properties have been seen as good candidates for random number generation for security keys, stream ciphers and many other applications in cryptography. It has been proved that pseudo random bit can be generated with the proper algorithm and manipulations, like those with the Henon map in 2D [1], and in 1D with the RTCM [2]. Whatever the technique been used, one of the crucial steps is the process of passing from floating-point numbers to bits or integers, that is quantization. It's a non-negotiable step and algorithms can fail to exhibit the required random properties when neglected or biased. Here we explore the general quantization techniques, their importance and a quick implementation in the standard 3D-Lorenz system.

2 Quantization in Chaos

In chaotic maps, the coordinates of the oribts (x, y and z in a 3-dimensional map) are in most cases continuous floating-point numbers. However, when the purpose is to be used in a Pseudo Random Number Generator (PRNG), a transformation must occur : the ouput is expected to be bits (0s and 1s) or integers. Quantization can be defined as the mapping function :

$$Q : \mathbf{R^n} \to (0,1)^m$$

n being the size of the random number generaed and m the size of the corresponding bits generated. There are many processes for quantization but, before talking about them, let's see why it's importance for quantization.

3 Core Importance of Quantization in Chaotic Systems

Despite being generated from chaotic maps, the random numbers generated are not "truly" random (as expected) and suffer issues. Fortunately, the step of quantization addresses to most of them.

3.1 The Correlation Problem

However chaotic the system is, at the end, it is still deterministic because it follows a precise formula. In the discrete implementation in the computer, the next state of the system depends on the previous one (we talk about iterations). It's obvious that the solutions are highly correlated, which is bad for any PNRG. So this is a problem that quantization try to correct by many means as we will see later.

3.2 Dynamic Degradation

This is a problem that every PRNG face. The map is continuous, and the problem is that computers can't represent correctlty high numbers, especially floating-point numbers. The reason being that computers don't have infinite precision, they are limited (64 bits in most cases). And at a time, we'll enter a limit cycle : we say the algorithm becomes *periodic*. And the numbers will start repeating themselves. This problem is ultimately unavoidable, and quantization techniques aim to delay this cycle to the maximum extent possible.

3.3 The Bias Trap

In chaotic maps that have more than one attractor (like Lorenz's system), the orbit can spend more time in a set than in the others. And when setting a threshold value for bits generation (as we'll see later), if not well selected, we can introduce bias in the selection, and bias means predictability. The selecion is meant to be uniform, but when the values stays in a set 51 % of the time (in a 2-attractors chaotic map), that means bias : 1% bias in cryptography is a dangerous vulnerability and quantization techniques have to lower this threshold to the greatest extent.

4 Common Quantization Techniques

The recent decade has seen many improvements in the field of chaos-based cryptography, in which there is a crucial need for random numbers generation. As such, many quantization techniques have been developed and the most common and robust ones will be presented below.

4.1 Bit Extraction (The Mantissa Slice)

This technique treats the floating-point chaotic state as a raw bit-string. Rather than using the entire value, we extract a specific segment of the binary representation. The most significant bits (MSBs) represent the predictable structure of the attractor, while the least significant bits (LSBs) represent high-entropy fluctuations. The generalized formula for extracting k bits after a shift of m positions is:

$$B = \lfloor |x| \cdot 2^m \rfloor \pmod{2^k}$$

with

- $|x|$ the absolute value of the state variable
- 2^m the magnification factor: a scaling constant where m is the number of bits the binary point is shifted to the right. It's common to see 10^m sometimes as a magnification factor
- $\lfloor ... \rfloor$ the floor function so we get an integer
- $\pmod{2^k}$ the modulo operator: the remainder of the division by 2^k, where k is the number of bits we want to keep. It acts as the final "slice", and cuts of the high value bits from the integer we got, and returns the k lowest bits which are the most random. That's because the higher ones still have the memory of the state of the chaos map.

By choosing a large m (e.g., $m = 32$), we "magnify" the decimals, pushing the chaotic noise into the integer range where it can be masked and used as a random block.

4.2 Symbolic Dynamics and Generating Partitions

As detailed by Stojanovski and Kocarev [3], quantization can be viewed through the lens of symbolic dynamics. We divide the state space into non-overlapping regions $\mathcal{C}_i$. For a 1D map or a single variable of the Lorenz system, a binary partition $\beta = \{\mathcal{C}_1, \mathcal{C}_2\}$ is defined:

$$S_n = \begin{cases} 1 & \text{if } x_n > \tau \\ 0 & \text{if } x_n \leq \tau \end{cases}$$

where τ is the threshold (often the median of the attractor). Stojanovski proves that if the partition is "Markov," the resulting sequence behaves as a Markov information source, allowing for a rigorous mathematical proof of entropy generation.

4.3 Chaos-Based Extremum Coding

A novel approach proposed by Araki et al. (2024) [4] involves sampling bits based on the local extrema of the chaotic waveform. Instead of sampling at fixed time intervals, the algorithm detects the peaks and troughs of the signal.

1. Detect a peak or trough in the discretized state $y(kT)$.
2. Compare the current extremum with the previous one.
3. Generate a Random Extremum-Coded Sequence (RECS).

This method is particularly robust against dynamic degradation because it uses the generated bits to dynamically modulate the system's own parameters, creating a self-correcting chaotic loop.

4.4 Post-Processing and Redundancy Reduction

Even with good quantization, bitstreams often retain residual correlations. Stojanovski (Part II) [5] outlines two main refinement methods:

- **Bit Skipping:** Retaining only every p-th bit to ensure temporal independence.
- **Bit Counting (XOR Mixing):** Grouping p bits into a block and performing a bitwise XOR:

$$Y_i = \bigoplus_{j=0}^{p-1} X_{ip+j}$$

XOR (Exclusive OR) mixing is generally superior as it significantly reduces bias and maximizes the entropy per output bit.

5 Implementation in the Lorenz System

To demonstrate these principles, we implement a quantization from the solutions of the Lorenz system:

$$(L_S)\begin{cases}\dot{x} &= \sigma(y-x)\\ \dot{y} &= x(\rho-z)-y\\ \dot{z} &= xy-\beta z\end{cases}$$

To a more in-depth understanding of the Lorenz system, please head to the dynamic study of Lorenz's System here.

To concretely demonstrate the quantization principles discussed above, we implement a full simulation of the Lorenz system coupled with a robust quantization chain. The numerical integration is performed with a fourth-order Runge–Kutta (RK4) scheme using a fixed time step $dt = 0.01$. The system parameters are set to the classical chaotic regime: $\sigma = 10, \rho = 28, \beta = 8/3$. The initial condition is $(x_0, y_0, z_0) = (1.0, 0.0, 0.0)$ and the trajectory is computed for $N = 10,000$ steps.

5.1 Quantization Algorithm

The raw floating-point states (x_i, y_i, z_i) are transformed into a high-entropy 8-bit integer stream through the following sequence:

1. **Magnification**: Each state variable is multiplied by 10^{14}. This large scaling factor pushes the microscale chaotic fluctuations into the integer domain, effectively amplifying the entropy contained in the least significant digits.

$$X_i = \lfloor |x_i| \cdot 10^{14} \rfloor, \quad Y_i = \lfloor |y_i| \cdot 10^{14} \rfloor, \quad Z_i = \lfloor |z_i| \cdot 10^{14} \rfloor$$

2. **LSB extraction**: The 8 least significant bits are retained from each magnified integer using a bitwise AND operation.

$$x_i^{lsb} = X_i \ \& \ 0\text{xFF}, \quad y_i^{lsb} = Y_i \ \& \ 0\text{xFF}, \quad z_i^{lsb} = Z_i \ \& \ 0\text{xFF}$$

3. **XOR mixing**: The three 8-bit blocks are combined via bitwise exclusive-or to produce the final quantized output Q_i.

$$Q_i = x_i^{lsb} \oplus y_i^{lsb} \oplus z_i^{lsb}$$

   This step is a direct application of Stojanovski's post-processing recommendation: it reduces any residual cross-correlations among the state variables and flattens the output distribution.

The resulting sequence $Q_i \in [0, 255]$ can be used either as a stream of 8-bit random integers or further decomposed into a raw binary bitstream by concatenating the binary representations of each Q_i.

5.2 Validation Through Visual Transformations

The four plots presented below directly compare the raw chaotic dynamics with the quantized output. Each pair of subfigures illustrates how the quantization process successively destroys the deterministic structure and yields a uniform, uncorrelated random source.

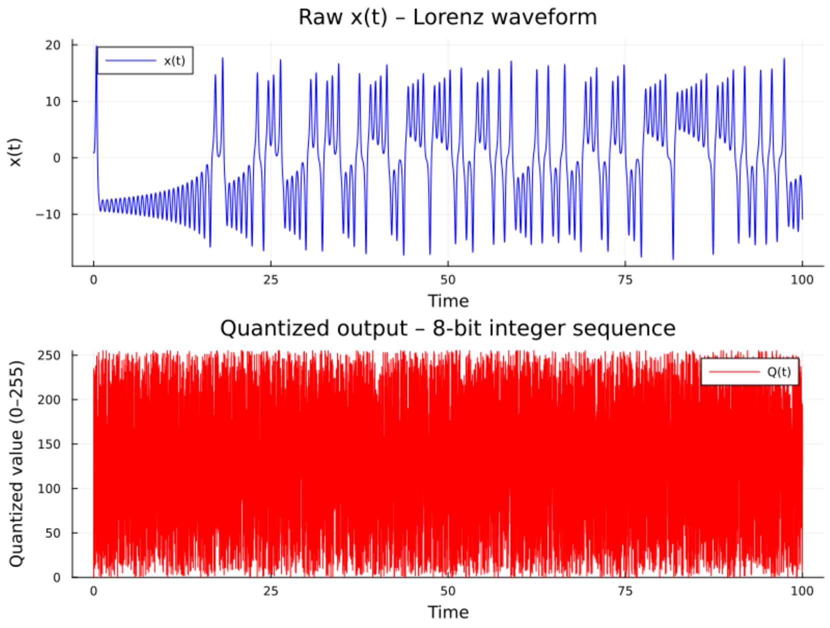

Figure 1: Waveform transformation. *Top:* raw $x(t)$ showing the smooth, oscillatory butterfly waveform characteristic of the Lorenz attractor. *Bottom:* quantized 8-bit integer output $Q(t)$. The continuity is completely lost; the plot resembles a random square wave with no discernible pattern in pulse widths.

5.3 Discussion of Results

The sequence of visualizations provides a rigorous, multi-perspective validation of the quantization algorithm.

- **Waveform** and **return map** confirm the destruction of short-term predictability.

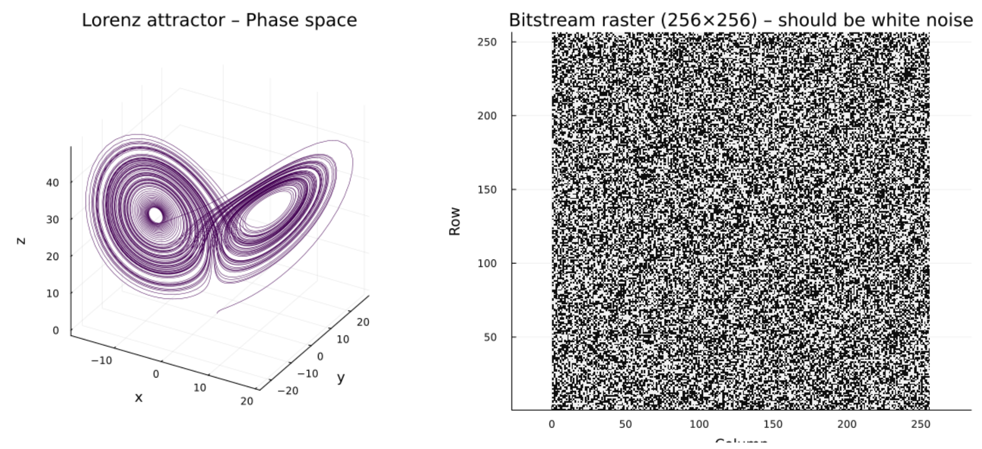

Figure 2: Phase space vs. bitstream raster (the “snow” test). *Left:* the classic Lorenz butterfly attractor in (x, y, z) phase space. This geometric object represents the deterministic information that must be hidden. *Right:* a 256×256 black-and-white raster of the first 65,536 bits of the generated bitstream. The absence of any lines, clouds, or large-scale patterns confirms that the quantization has successfully transformed the structured chaotic signal into pure white noise.

- The **raster image** demonstrates that the spatial distribution of bits is indistinguishable from ideal white noise.
- The **autocorrelation** measurement gives a quantitative proof that the temporal correlations inherent to the Lorenz system are eliminated.

Taken together, these results indicate that the simple combination of magnification, LSB extraction, and XOR mixing is sufficient to turn the deterministic Lorenz flow into a high-quality pseudo-random number generator.

6 Conclusion

This study has demonstrated the critical necessity of a robust quantization pipeline in transforming continuous chaotic flow into cryptographic entropy. By implementing a sequence of magnification (10^{14}), LSB extraction, and XOR-based mixing on the Lorenz system, we successfully converted deterministic trajectories into a statistically independent bitstream.

Our visual diagnostics confirm the effectiveness of this approach: the collapse of the autocorrelation function into a delta spike and the transition from structured phase-space attractors to uniform ”white noise” rasters prove

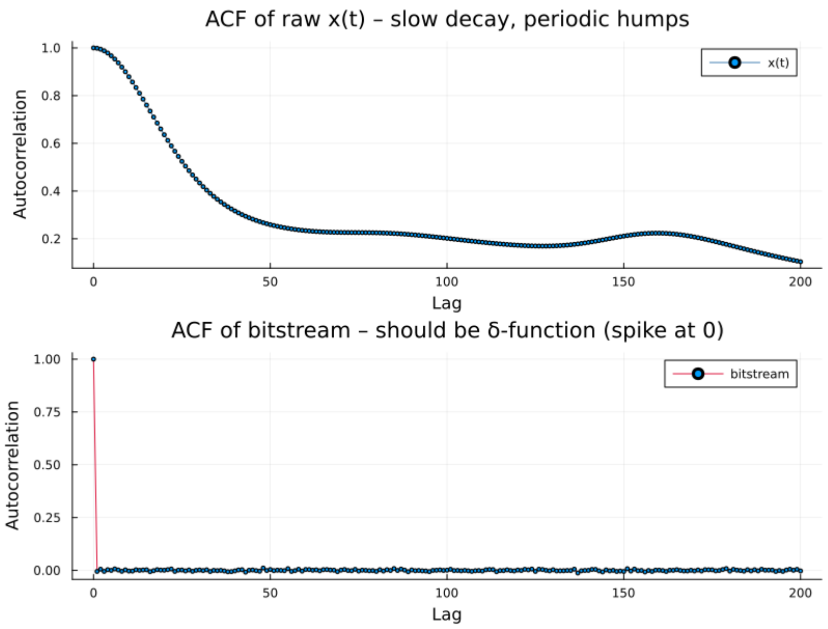

Figure 3: Autocorrelation function (ACF) transformation. *Top:* ACF of the raw x variable. The slow decay and periodic humps reveal long-range memory and deterministic periodicity. *Bottom:* ACF of the final bitstream. The function is a perfect delta spike at lag zero and fluctuates within ± 0.02 elsewhere. This proves that the quantization has destroyed all linear correlations.

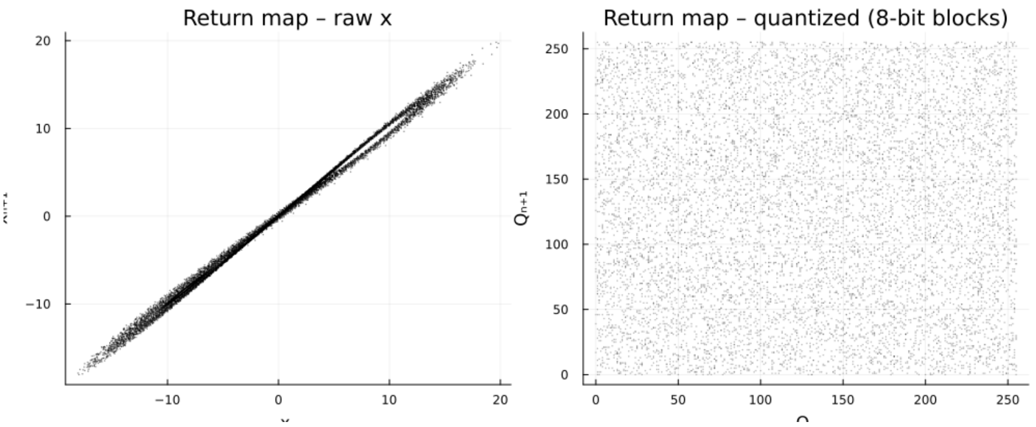

Figure 4: Return maps (x_n vs. x_{n+1}). *Left:* return map of the raw x variable. The thin, well-defined curve is the deterministic "logic" of the Lorenz generator. *Right:* return map of the quantized 8-bit blocks Q_n. The points uniformly fill the entire 256×256 square without any visible structure.

the elimination of the *correlation problem*. Furthermore, the use of LSB extraction and multi-variable mixing effectively mitigates both the *bias trap* and *dynamic degradation* inherent to digital chaotic implementations.

While the current RK4-based simulation provides high-quality randomness for a $10,000$-step sequence, future work can possibly focus on:

1. **Statistical Certification:** Subjecting the bitstream to the full NIST SP 800-22 and Dieharder test suites.
2. **Throughput Optimization:** Investigating Chaos-Based Extremum Coding as a higher-rate alternative to XOR mixing.
3. **Hardware Generalization:** Porting the quantization framework to hyperchaotic maps and FPGA-based embedded systems.

The simplicity and demonstrable effectiveness of this quantization chain suggest that chaos-based PRNGs, when correctly engineered, are highly viable for lightweight security applications and secure data transmission.

References

[1] M. Suneel, "Cryptographic pseudo-random sequences from the chaotic henon map," 2006.

[2] M. Irfan and M. A. Khan, "Cryptographically secure pseudo-random number generation (cs-prng) design using robust chaotic tent map (rctm)," 2024.

[3] T. Stojanovski and L. Kocarev, "Chaos-based random number generators part i: Analysis," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 48, no. 3, pp. 281–288, 2001.

[4] S. Araki, J.-H. Wu, and J.-J. Yan, "A novel design of random number generators using chaos-based extremum coding," *IEEE Access*, vol. 12, pp. 24039–24053, 2024.

[5] T. Stojanovski, J. Pihl, and L. Kocarev, "Chaos-based random number generators part ii: Practical realization," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 48, no. 3, pp. 382–388, 2001.