Grenze International Journal of Engineering and Technology, June Issue

Empirical Profiling of Noise Growth in the BGV Scheme using Lattigo v6

Gokul S Unnikrishnan[1], Somnath Sinha[2] and Binayak Dutta[3]
[1-3]Department of Computer Science, CHRIST University, Bengaluru, India
Email: gokul.unnikrishnan@mca.christuniversity.in, somnath.sinha@christuniversity.in, binayak.dutta@christuniversity.in

***Abstract*— This paper presents an empirical study on how noise grows in the Brakerski-Gentry-Vaikuntanathan (BGV) homomorphic encryption scheme using the Go-based Lattigo v6 library. To bridge the gap between worst-case theoretical bounds and practical behavior in a Residue Number System (RNS) implementation, noise evolution is studied across ring degrees ranging from N=4096 to N=32768. The results indicate that ciphertext multiplication combined with relinearization is the largest contributor to noise amplification, consuming approximately 29.5 bits of the available noise budget per operation, whereas addition consumes exactly one bit. Furthermore, this study supports the critical role of modulus switching in maintaining a sufficient noise margin. Based on these observations, a library-specific design heuristic is proposed: allocating approximately 30 bits of modulus capacity per multiplicative level. This provides a concrete reference point for engineering decisions.**

***Index Terms*— Homomorphic Encryption, BGV, Noise Growth, Lattigo, RNS, Modulus Switching, Parameter Selection, Cryptographic Engineering.**

I. INTRODUCTION

Homomorphic Encryption (HE) represents a paradigm shift in secure computation, enabling arbitrary computation on data while it remains encrypted. The algebraic structure of plaintexts—typically mapped to polynomial rings—allows operations performed on ciphertexts to translate directly to operations on the underlying data. This capability is critical for privacy-preserving applications in untrusted environments, such as secure cloud computing, genomic data analysis, and privacy-preserving machine learning inference.

However, the transition of HE from a hypothetical model to practical implementation is hindered by high computational costs and the complexity of cryptographic parameter management. Unlike static encryption schemes (e.g., AES or RSA), HE ciphertexts are dynamic entities that degrade in quality with every arithmetic operation.

A. The Dilemma of Noise Accumulation

In Lattice-based HE constructions, including the Brakerski-Gentry-Vaikuntanathan (BGV) scheme, noise accumulation is the primary bottleneck. BGV security relies on the hardness of the Ring Learning With Errors (RLWE) problem, where a small, random error term is injected during encryption.

This noise is cumulative and non-linear. While homomorphic addition increases noise linearly, homomorphic multiplication causes it to grow polynomially, primarily due to the tensor product of ciphertexts and the subsequent key-switching operations.

Grenze ID: 01.GIJET.12.2.543

If the noise magnitude exceeds a specific threshold defined by the current ciphertext modulus, the decryption process fails, rendering the data irretrievable. To mitigate this, BGV employs modulus switching, a technique that scales down the ciphertext modulus to reduce the absolute magnitude of noise, trading bit-level security for greater computational depth.

B. The Dislocation Between Theory and Implementation

While the theoretical bounds of BGV noise growth are well-documented, real-world performance often diverges due to implementation-specific choices. Modern HE libraries, including Lattigo, utilize a Full-RNS (Residue Number System) variant of BGV to avoid multi-precision arithmetic. This introduces specific noise terms related to the CRT decomposition base (gadgets) used during key switching.

Previous comparative studies have established that conventional heuristic bounds tend to be too loose for efficient parameter selection, necessitating empirical profiling. However, the \textbf{Go/Lattigo} library, noted for its high-performance integer arithmetic and concurrency, remains under-characterized. Theoretical models derived for C++ libraries (e.g., SEAL, HElib) cannot be directly applied to Lattigo's specific RNS optimizations.

C. Contributions

This study bridges the gap between theoretical noise bounds and engineering reality within the Lattigo ecosystem.

Our contributions are:

- Operation-Level Quantification: We quantify the noise budget consumption of atomic operations, identifying multiplication with relinearization as the dominant cost factor (approx 29.5 bits).
- Theoretical Divergence Analysis: We analyze the disparity between worst-case theoretical noise bounds (often >50 bits) and the observed average-case behavior, validating tighter parameter selection for production use.
- Design Heuristics: We derive a library-specific parameter selection guideline, proposing a 30-bit allocation rule per level to minimize trial-and-error in system configuration.

II. PRELIMINARIES

To contextualize the empirical results, we review the BGV scheme's mathematical structure, specifically focusing on the RNS variant used in Lattigo v6.

A. Ring-LWE and BGV Structure

The BGV scheme operates over the cyclotomic ring $R = \mathbb{Z}[X]/(X^N + 1)$, where N is a power of two. Ciphertexts are defined over the ring $R_Q = R/QR$, where Q is the ciphertext modulus. A ciphertext ct is a pair of polynomials $(c_0, c_1) \in R_Q^2$. Decryption is defined via the linear relation:

$$[c_0 + c_1 \cdots]_Q = m + v \pm odt \tag{1}$$

where s is the secret key, m is the message (in plaintext space R_t), and v is the error (noise) term.

B. RNS Representation and Gadgets

In Lattigo, the modulus Q is decomposable into a chain of coprime moduli $q_0, \ldots, q_L$. Operations are performed component-wise over small integers (CRT representation).

Key Switching Noise: Multiplication expands the ciphertext to three polynomials. Reducing this back to two requires relinearization, which introduces noise proportional to the gadget decomposition base w. The added noise term e_{relin} is bounded by:

$$|e_{relin}| \approx \frac{1}{2} \cdot \sqrt{d_{gadget}} \cdot w \cdot \sigma_{err} \tag{2}$$

where d_{gadget} is the number of decomposition levels. Our empirical profiling effectively measures the aggregation of this term in practice.

C. Noise Budget Metric

Correctness holds if the infinity norm of the error, $|v|_\infty$, is less than the decoding boundary $\lfloor Q/2 \rfloor$. We define the noise budget available at any stage logarithmically as:

$$Budget = log_2 Q_\ell - log_2 |v|_\infty \tag{3}$$

In a leveled scheme like BGV, Q_ℓ decreases as levels are consumed. The budget represents the remaining "room" for noise before it overlaps with the message bits.

III. Related Work

Singh et al. [1] conducted a detailed survey of Homomorphic Encryption, concluding that even though this technol-ogy enables computation to be carried out in a privacy-preserving way, it suffers from significant drawbacks: high computational overhead and the intricacies of managing noise, which prevent its broader use. Their article creates a baseline of motivation for our research by defining noise as a major obstacle to entry. However, being a high-level analysis, it does not provide engineering information on noise proliferation in modern software libraries, necessitat-ing the granular profiling performed herein.

Tsuji and Oguchi [2] compared various Fully Homomorphic Encryption schemes and libraries, concluding that im-plementation selection has a major effect on processing speed and throughput. Their results indicate the performance capability of the Go-based Lattigo library compared to its C++ competitors. Although their research focuses on exe-cution time, it is silent on the internal noise mechanics controlling correctness; this gap is filled by our definition of the noise cost for the operations they benchmarked.

Costache et al. [3] conducted a comparative study of noise growth in BGV and FV schemes. As observed, BGV tends to allow a higher circuit depth with a similar set of parameters but requires more complex modulus manage-ment. This offers a theoretical basis for choosing BGV in deep computational situations. Nonetheless, their empirical results were based on previous versions of HElib and SEAL, leaving a knowledge gap regarding the effect of con-temporary RNS optimizations in Lattigo v6 on these established noise profiles.

Surveying the usage of HE in machine learning applications within cloud and fog environments, Marcolla et al. [4] found that system architecture should support the latency caused by cryptographic expansion. This highlights the significance of effective parameter determination in distributed systems. However, they focus on system-level archi-tecture rather than the low-level arithmetic noise properties that ultimately determine whether a deployed circuit achieves correct behavior.

The OpenFHE library on GPU architectures was profiled by Papadakis et al. [5], who concluded that data marshal-ling overheads often exceeded arithmetic costs in hardware-accelerated environments. This emphasizes the need to profile implementation bottlenecks. Nevertheless, they investigated only GPU-based implementations within the OpenFHE ecosystem, which does not reflect the noise properties of CPU-bound execution in the Go environment where memory management differs.

Gouert and Mouchet [6] suggested systematic parameter selection for BGV, concluding that the trade-off between security bits and ring dimension is critical to maximize depth. This provides a conceptual scheme for constructing modulus chains. The limitation of their work is its reliance on general theoretical bounds, which we aim to make more specific by providing a real, library-specific constant—``bits-per-level''—based on actual execution.

Yuan and Gao [7] studied parameter tuning in resource-constrained scenarios, determining that optimizing polyno-mial moduli can result in significantly lower resource consumption. This is relevant for reducing the memory foot-print of Lattigo. However, their optimization focuses on storage and speed rather than fine-tuning the noise budget, which can result in parameters that are efficient but fragile without the empirical safety margins we introduce.

Murphy [8] applied the Central Limit Theorem to the noise analysis of Ring-LWE, concluding that error variables are sub-Gaussian and that average-case noise growth is significantly smaller than worst-case analysis. This theoretical understanding explains the witnessed safety margin in actual deployments. We connect this mathematical model with engineering reality by testing Murphy's probabilistic predictions against actual trace data in Lattigo.

Bai et al. [9] focused on gadget matrix optimization, concluding that specific gadget decompositions can optimize ciphertext expansion in BGV. Since relinearization noise is directly proportional to gadget size, their work is directly applicable to our results on multiplication costs. However, they focus on matrix construction theory, whereas we measure the real operational noise penalty of such gadgets during operation.

Kim et al. [10] suggested methods to generalize bootstrapping without plaintext modulus constraints, finding that this allowed computation beyond the leveled limit. This is the next logical step for circuits exceeding the depths we con-sider. Their work, however, focuses on the refresh procedure, whereas our work characterizes the leveled operations that necessitate bootstrapping, providing the data needed to determine when such expensive operations are actually required.

Hwang et al. [11] proposed a level-aware key-switching approach, concluding that key-switching parameters can be optimized according to the level to enhance efficiency. This is highly relevant since key switching is found to be the most prominent noise source. Their work suggests a new optimization but does not describe the baseline behavior of the standard key-switching algorithms currently deployed in Lattigo, which serves as the control group for such op-timizations.

Cheng and Shieh [12] developed a key-switching hardware architecture for the CKKS scheme, finding that special hardware could overcome the latency of high-degree polynomial multiplication. This supports our observation that multiplication and relinearization are the costliest processes. Nevertheless, their focus on CKKS and ASIC/FPGA hardware leaves a gap in the noise characterization of BGV in software-only implementations, which remain the standard for accessible development.

IV. METHODOLOGY

A. Implementation Enviroment

The test-based system is built around the Go programming language version 1.22 and the Lattigo library, version 6. Lattigo was chosen due to its optimum full-RNS implementation. The experiments on the workstation were conduct-ed on the following hardware and software specifications:

- CPU: AMD Ryzen 5 3550H (4 cores, 8 threads, 2.10GHz).
- RAM: 16 GB DDR4 memory.
- OS: Linux (kernel 6.17.8), running on a custom Arch-based distribution.
- Concurrency: To impose deterministic ordering and isolate the increase in crypto-graphic noise due to artifacts caused by operating-system scheduling, the evaluation harness was constrained to single-threaded execution.

B. Experimental Design

We constructed a modular test harness to generate BGV circuits with varying multiplicative depths.

Parameter Generation

Cryptographic parameters were generated to ensure a minimum of 128-bit security according to the Homomorphic Encryption Standard. We evaluated ring degrees $N \in \{4096,8192,16384,32768\}$. For each N, the library constructs a compliant modulus chain $Q_L = \prod_{i=0}^{L} q_i$, where each $q_i \approx 50$ to 60 bits depending on the specific Lattigo preset.

Input Encoding

Plaintext inputs were sampled according to a uniform distribution with constant density, in order to induce worst-case noise behavior: U[0,t). This is the best way to maximize the Hamming weight of the encoded polynomial and ensure that measurements of the noise induced by it are not an optimistic sparse case but an adversarial usage scenario.

C. Noise Profiling Routine

Noise accumulation was measured using a deterministic evaluation loop (Algorithm 1). This procedure isolates per-operation noise growth by tracking the invariant noise budget after every arithmetic step.

Algorithm 1 Noise Profiling Routine

Input: Ring degree N, Plaintext modulus t
Output: Noise budget log $\mathcal{L}$
$sk, pk \leftarrow \text{KeyGen}(N)$
$ct \leftarrow \text{Encrypt}(pk, \mathcal{U}[0,t))$ {Dense packing}
while $\text{Level}(ct) > 0$ **do**
  $\beta \leftarrow \text{MeasureInvariantNoise}(ct, sk)$
  $ct \leftarrow \text{Mul}(ct, ct)$ {Squaring}
  $ct \leftarrow \text{Relinearize}(ct)$
  if Leveled Mode **then**
    $ct \leftarrow \text{Rescale}(ct)$
  end if
  $\mathcal{L}.\text{append}(\beta, \text{Level}(ct))$
end while

D. Data Acquisition

Telemetry module logs the state of ciphertext after each cryptographic operation. The noise budget is then obtained by decrypting the in-between ciphertext to obtain the error vector v.

$$Budget = LevelModBits - log_2(|v|_\infty) \quad (4)$$

TABLE I: INITIAL NOISE BUDGET AT MAXIMUM START LEVEL

N	Start Level	Budget (Bits)	Samples
4096	1	14.56	99
8192	4	31.69	342
16384	9	62.33	747
32768	19	119.62	1557

TABLE II: INITIAL NOISE BUDGET AT MINIMUM START LEVEL

N	Start Level	Budget (Bits)	Samples
4096	0	4.27	90
8192	0	2.12	306
16384	0	2.54	666
32768	0	2.30	1386

V. RESULTS

A. Homomorphic Multiplication

The most common noise amplification is ciphertext multiplication and relinearization. A single multiplication uses up about 29.5 bits of the noise budget as shown in Table 3. This cost is also largely independent of the parameter N which implies that the overhead of relinearization which is especially the noise of gadget decomposition in key switching will dominate the consequences that can be ascribed to the size of the polynomial degree.

TABLE III: INITIAL NOISE BUDGET AT MINIMUM START LEVEL

Operation	Cost (Δ bits)	Behavior
Mult. & Relinearization	≈29.5	Dominant, Stochastic
Homomorhpic Addition	1.10	Deterministic, Linear

B. Homomorphic Addition

Homomorphic addition shows minimal overhead. Self-addition increases the noise magnitude by exactly one bit, consistent with the doubling of the error norm ($|v + v| \leq 2|v|$). This disparity is visualized in Fig. 1.

Fig. 1: Comparative analysis of average noise consumption. Ciphertext multiplication (with relinearization) consumes 29.5 bits, whereas self-addition consumes exactly 1 bit

C. Effect of Modulus Switching

The interplay between noise growth and modulus management is captured in Fig. 2.

- Sawtooth Recovery (Leveled): The noise budget takes the form of a "sawtooth" pattern when rescaling is on. The sharp fall caused by multiplication is then followed by a partial recovery caused by rescaling. This recovery occurs because dividing the ciphertext by a prime q_i scales down the error term v proportionally ($v' \approx v/q_i$), effectively resetting the noise-to-modulus ratio.

- Monotonic Decay (Unmanaged): Disabling rescaling results in strictly monotonic decay. Fig. 2 shows that noise builds up very quickly along the red dashed curve, thus causing instantaneous decryption failure in shallow modulus chains.

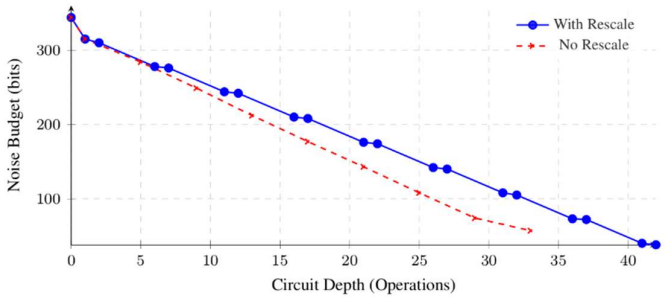

Fig.2: Evolution of the noise budget (N = 16384). The "sawtooth" pattern (blue) shows budget recovery via rescaling, while the dashed line (red) shows rapid linear decay without it

D. Impact of Ring Degree

The degree of the polynomial ring can be increased to allow the creation of longer modulus chains and the initial noise budget can be raised, but at an extremely high cost in terms of computational complexity. The depths achievable have been listed in Table 4. Specifically, with a ring degree of N=32768, the scheme attains a median depth of 9.5 multiplicative gates and thus it supports the idea that the Lattigo implementation scales to circuits that have medium depth.

TABLE IV: ACHIEVED MULTIPLICATIVE DEPTH BY RING DEGREE

N	Rescale	Median Multiplication
4096	Yes/No	0.5
8192	Yes/No	2.0
16384	Yes/No	4.5
32768	Yes/No	9.5

VI. ANALYSIS AND DESIGN IMPLICATIONS

A. Theoretical vs. Empirical Divergence

Theoretical studies of multiplication in the Brakerski-Gentry-Vaikuntanathan (BGV) scheme are usually expected to amplify worst-case error on the order of $B_{mult} \approx \delta_R \cdot B^2$, where δ_R is the multiplicative expansion factor of the underlying ring. These worst-case estimates under representative parameter choices (namely $N = 16384$) suggest that the noise budget loss at each multiplication level will exceed 50 bits.

We find in our own empirical measurements that the average noise budget consumption of approximately 29.5 bits is significantly below these theoretical limits. This discrepancy highlights the safety margin inherent in the stochastic nature of Ring Learning-with-Errors (RLWE) noise. Designs that are based on such conservative worst-case bounds are therefore likely to be over-parameterized, incurring unnecessary computational costs. The implementation of Lattigo, which uses effective gadget techniques, successfully maintains noise growth within average-case predictions.

B. The 30-bit Partitioning Heuristic

Based on the data, we propose a design heuristic where the required total modulus size $log\ Q$ is estimated as:

$$log\ Q \approx 30 \cdot L + \beta_{margin} \quad (5)$$

where L is the desired multiplicative depth and β_{margin} accounts for encoding and final decryption noise. This heuristic serves as a substantive parameter generation initial anchor, which replaces abstract boundaries by an empirically obtained parameter.

C. Budget Management

The configurations that have a low modulus depth are characterized with severe degradation (Fig. 2). This means that a multiply then rescale approach is thus necessary to ensure maximum circuit efficiency. In spite of the fact

that theoretical BGV can be rescaled in due time, experimental results have shown that rescaling in real-time maintains a more desirable noise margin and allows a much tighter choice of parameters, thus preventing the rapid saturation that occurs in unregulated traces.

D. Practical Deployment Steps

These findings enable a deterministic parameter-selection strategy for Lattigo developers:

- Estimate the required multiplicative depth L of the circuit.
- Allocate ≈30 bits per level plus a safety margin (e.g., 20 bits).
- Select the smallest ring degree N capable of supporting the resulting total modulus log Q while satisfying 128-bit security standards.

V. CONCLUSION

This work derives a conclusive noise profile of the Brakerski-Gentry-Vaikuntanathan (BGV) scheme in Lattigo v6. Isolating the cost of atomic operations enabled us, with some overhead of around 29.5 bits, to compute an estimate of an overhead due to multiplication and relinearization. On these lines, we suggest that a system architect should allot about 30 bits of modulus capacity to each unit of multiplicative depth. Whereas the additive noise contribution, approximately one bit, can be ignored, linear accumulation trees of large depth still must be considered.

The strong disparity between the sawtooth recovery typical of leveled mode and the fast attenuation typical of unmanaged mode illustrates the necessity of having modulus switching in any non-trivial computation. Future research will generalize this profiling to the BGV bootstrapping process, which will make it possible to permit better homomorphic encryption in the Lattigo model.

ACKNOWLEDGMENT

The authors wish to thank the Department of Computer Science at Christ University for supporting this research.

REFERENCES

[1] V. K. Singh, A. S. Chauhan, A. Singh, and R. Thakur, “Homomorphic Encryption: Hands Inside the Gloves,” in Proc. Int. Conf. on Computing, Dec. 2023, pp. 248--253.
[2] A. Tsuji and M. Oguchi, “Comparison of FHE Schemes and Libraries for Efficient Cryptographic Processing,” in Int. Conf. on Computing and Networking, Feb. 2024.
[3] C. Costache, D. Micciancio, and T. Hoang, “Empirical Comparison of Noise Growth in the BGV and FV Fully Homomorphic Encryption Schemes,” in Selected Areas in Cryptography (SAC 2019), Dec. 2019, pp. 307--328.
[4] C. Marcolla, A. Cammarota, A. G. D. Ferrari, C. D'Antini, R. Falcone, and T. Hoang, “A Survey on Practical Homomorphic Encryption Deployment for ML in Fog/Cloud,” IEEE Trans. Cloud Comput, pp. 1--19, Jun. 2022.
[5] M. Papadakis, L. Akrouh, H. Savvides, and C. C. Lo, “GPU Accelerated BGV Homomorphic Encryption Parameter Profiling in OpenFHE,” in 2024 IEEE Int. Parallel and Distributed Processing Symp. (IPDPS), May 2024, pp. 198--209.
[6] S. Gouert and C. Mouchet, “Parameter Selection Guidelines for BGV Based Fully Homomorphic Encryption Schemes,” IEEE Access, vol. 8, pp. 185326--185339, 2020.
[7] J. Yuan and F. Gao, “Efficient Parameter Tuning for BGV Homomorphic Encryption Schemes under Resource Constraints,” Journal of Cryptographic Engineering, vol. 10, pp. 57--70, 2020.
[8] V. Murphy, “A Central Limit Approach for Ring-LWE Noise Analysis,”' Cryptology ePrint Archive, Report 2024/665, July 2024.
[9] S. Bai, A. Chen, J. Hao, M. Hoang, and W. Li, “Optimizing Gadget Matrix and Ciphertext Size for BGV/GSW Homomorphic Encryption,” in Int. Workshop on Optimization in Cryptography, July 2024.
[10] H. Kim, J. Lee, and D. Park, “General Bootstrapping for Lattice-Based FHE Without Plaintext-Modulus Constraint,” in Int. Conf. on Cryptology and Network Security, May 2024, pp. 108--127.
[11] S. Hwang, J. Kim, and S. Lee, “Level-Aware Key-Switching Method for Efficient Homomorphic Encryption,” IEEE Trans. on Information Forensics and Security, vol. 18, pp. 1296--1308, March 2023.
[12] Y. Chen and T. Shieh, “Design of Area-Efficient Key Switching Architectures for CKKS Homomorphic Encryption,” Sensors, vol. 23, no. 10, p. 4594, April 2023.