Homomorphic Encryption: Evaluating CKKS and BGV in the Modern Era of Secure Computation

Debbie Thai[*] Ujjwal Baranwal[†] Sunil Chowdary Vejendla[‡]

Abstract

Homomorphic encryption (HE) schemes work to ensure data privacy and security in various applications. We address the lack of a universally accepted HE scheme and present a comparative analysis between the fully homomorphic encryption (FHE) schemes CKKS (Cheon-Kim-Kim-Song) and BGV (Brakershi-Gentry-Vaikuntanathan), with a focus on their efficiency and usability. Our results revealed nuanced differences in the usability and efficiency of both schemes highlighting their similarities, strengths, and weaknesses. Overall both CKKS and BGV have comparable efficiency, security, and precision performance, with CKKS being particularly useful for approximate arithmetic in machine learning applications.

1 Introduction (20 points)

Over the years, various homomorphic encryption (HE) schemes have evolved to ensure the confidentiality and integrity of sensitive data. The abundance of different schemes raises a challenge in determining the efficiency, interoperability, and standardization of homomorphic encryption. Overall, the lack of a universally accepted homomorphic encryption scheme causes complications in the integration of secure computing technologies across different platforms and applications. This report aims to conduct a comprehensive comparison between CKKS (Cheon-Kim-Kim-Song) and BGV (Brakerski-Gentry-Vaikuntanathan), two prominent fully homomorphic encryption schemes. The goal of the comparison is to evaluate and conclude which model displays optimal efficacy, precision, and security in real-world applications. The study's motivation stems from homomorphic encryption's unique ability to allow "computing over encrypted data without decrypting them first"[2]. This ability is crucial for ensuring sensitive information remains confidential especially in organizations where privacy is essential. Homomorphic encryption enables secure outsourcing of computations, facilitates secure data sharing, and preserves data privacy. All these benefits have been met with a growing demand for an efficient and scalable HE scheme. BGV and CKKS meet these requirements, with each offering a unique set of strengths and weaknesses. Derived from the Ring Learning with Errors (RLWE) problem, CKKS enables computation over encrypted data with an added ability to work with approximate arithmetic operations. BGV incorporates lattice-based cryptography, allowing computations on encrypted data with a high level of security. The papers reviewed in our research explored the characteristics of both encryption schemes, detailing their limitations and capabilities. We wanted to put together a comprehensive and up-to-date comparative analysis that discusses the differences in choosing between CKKS and BGV. This investigation will aid in the ongoing improvement of homomorphic encryption, specifically to address the difficulties in selecting the most suitable scheme when it comes to computational complexity and operational efficiency.

2 Problem Setup and Threat Model (20 points)

Partial Homomorphic Encryption (PHE) Schemes are schemes that support either additive or multiplicative operations on encrypted data, allowing computation on the encrypted data without the need for decryption. Fully Homomorphic Encryption (FHE) goes a step further than Partial Homomorphic Encryption, allowing both additive and multiplicative operations on encrypted data. This, while secure, is frequently slower due to more complex calculations. So ever since the development of the first Fully Homomorphic Encryption Scheme by Craig Gentry, there have been significant advancements in the field. These improvements aimed to not only enhance the efficiency and security of the scheme but also broaden its applicability in various domains[1]. The variety of homomorphic schemes developed, lead to the observation of different behavior from different schemes such as DGHV, BGV, BFV, CGGI, CKKS, and GSW implying that there are certain criteria for using a specific scheme.

[*] CSE Master's 1st year

[†] CSE Bachelor's 4th Year (Enrolled in 4+1 Program)

[‡] CSE Master's 1st year

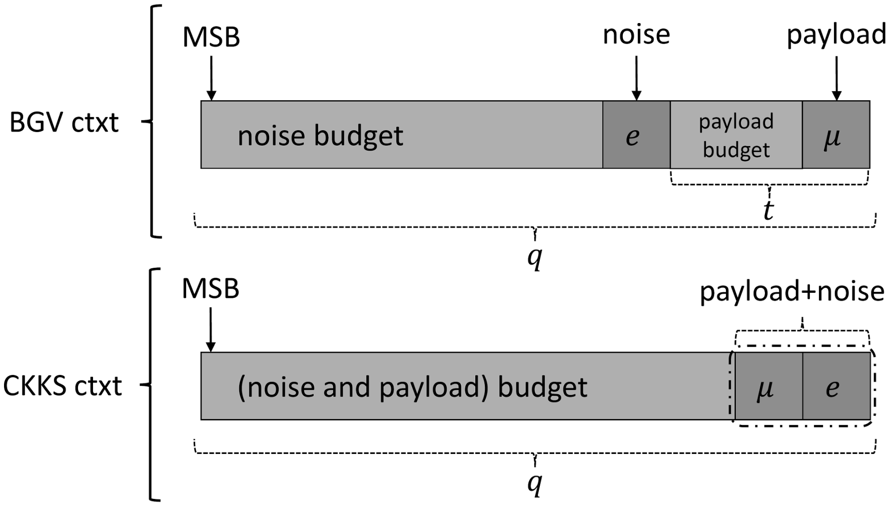


Figure 1: BGV vs CKKS Cipher Text Structure

Problem Setup: This research aims to conduct a comprehensive comparison on these criteria for the CKKS and BGV homomorphic encryption schemes. The comparison will focus on computational efficiency and usability. The goal is to provide insights into the strengths and weaknesses of each scheme and guide their application towards a secure environment.

3 Theory/Construction/Analysis (30 points)

Cheon et al. set out to create homomorphic encryption for approximate number arithmetic addressing the significant loss of precision in current FHE methods. CKKS introduces encryption noise added to plaintext during encryption, with a decryption structure with the form $\langle c, sk \rangle = m + e(mod \quad q)$, for some small error e [3]. The encryption error is used to enhance the security of the encryption and is treated as errors from approximate calculations. The output of the decryption algorithm is regarded as an approximate value with high precision, highlighting the presence of errors. This process of introducing errors proves to minimize the impacts of errors on the accuracy of scheme results alongside the rescaling procedure. The procedure reduces the size of ciphertext allowing for faster computation over large integers. Description of CKKS encoding, decoding, and rescaling is shown in (Fig.2). The rescaling procedure and encryption noise management ensure that errors introduced during homomorphic operations remain under control and do not affect the accuracy of the result, which allows CKKS the ability to handle floating-point approximation [3]. The ability to ensure high precision during computations makes it useful for machine learning applications. While the security of CKKS is strengthened from its use of encryption noise and rescaling, Li & Micciano exploit a vulnerability that occurs when the decryption algorithm outputs $m + e$ [6]. The researchers use the Linear Key-Recovery Attack to successfully recover the secret key in the CKKS scheme with high probability. Despite this, the attack is only effective under certain circumstances and CKKS's overall introduction of errors and error-tolerant techniques balance the security and computational efficiency of the scheme. [6] also provides simple countermeasures to heighten CKKS security, namely removing the decryption output feature. CKKS also employs modulus switching and parameter customization for dynamic adjustment of security level and efficiency, and user ability to modify the scheme for certain use cases. Current research conducted on various bootstrapping techniques shows promise in improving the efficiency and precision of CKKS as well.

Brakerski-Gentry-Vaikuntanathan (BGV) scheme is a Fully Homomorphic Encryption (FHE) cryptosystem proposed in 2011 by Brakerski et al. This FHE cryptosystem is based on the Ring Learning with Error (RLWE)[4] model which serves as the foundation of new cryptographic algorithms used to protect against Quantum Cryptanalysis. The main strength of this scheme is to work effectively with integer values. This is because the BGV scheme operates in the ring of integers, and the encryption and decryption processes are designed to handle integer values most optimally. BGV Schemes are initialized with integer parameters; m

(cyclotomic field), t (plaintext modulus) and, q (coefficient modulus) such that the greatest common divisor of t and q is 1 ($GCD(t, q) = 1$). The common quotient rings notation for BGV is $R = Z[x]/(\phi_m(x))$, $R_t = R/tR$, and $R_q = R/qR$[2]; where $\phi_m(x)$ denotes the m-th cyclotomic polynomial, *m(x) (polynomial)* is the message present as an element in R_t and s is the secret-key present as an element in R_q. The parameters and the rings are strategically chosen in a way that balances security & error margin for the computations and helps in the efficient computation of integers. The Decryption formula for BGV begins with a dot product of the secret-key (s) vector and the cipher-text pair (c_0, c_1) vector: $c_0 + c_1 s = m(x) + tv + \alpha q$ With the decryption function returning the plain-text message as: $m(x) = ((c_0 + c_1 s) \bmod q$. Additionally, under certain conditions, BGV is an effective scheme for Machine Learning over Encrypted Data. Most machine learning models necessitate bootstrapping (a technique used by multiple HE that allows a party to refresh the encrypted data reducing noise) because they require a deep circuit. The Ring Model in BGV provides efficient and secure calculation which Machine Learning Models can conveniently use for further calculation and analysis/predictions. BGV also supports modulus switching, a technique which scales the ciphertext (c_0, c_1) with modulus q to (c_0', c_1') with modulus q'. This is effectively used with bootstrapping to reduce noise[2].

4 Evaluation (20 points)

Since CKKS is partially based on BGV and its Ring Learning with Error (RLWE) model, there are multiple similarities between them. They have a similar structure that allows parameters to customize the encryption scheme. CKKS uses parameters p (RNS Number), t (plaintext modulus), and q (ciphertext modulus) which is very similar to BGV parameters m (degree of polynomial), t (plaintext modulus), and q (ciphertext modulus). These parameters are responsible for efficient and secure encryption/decryption personalized schemes. Since they both are based on a similar structure and model, they are also believed to be Quantum Resistant. However, there are prominent differences between the two models which makes each of them have a unique application. BGV is a Lattice-based cryptography scheme that is closely related to Ring LWE which makes the BGV scheme rigid in its application. BGV is thus better suited for integers. CKKS is derived from BGV and its RLWE model but it is modified to an Approximate Number System (ANS) model. This makes CKKS efficient in calculations with real and floating point numbers. CKKS and BGV are both suitable schemes for Machine Learning over Encrypted Data, but since they have different strengths, BGV performs better with integer data while CKKS handles real and floating point numbers more efficiently.

- As shown in Figure 1, the ciphertext structure for BGV is different from CKKS. BGV scheme handles the noise budget, noise and payload differently and focuses on noise budget as its Most Significant Bit. On the other hand, CKKS handles (noise and payload) budget and payload+noise together. CKKS chooses (noise and payload) budget as its Most Significant Bit.
- As shown in Figure 3[5], the chart compares the computation_time and the polynomial degree n up to 10 for four different schemes. Since the values used for this weren't strictly integers, one can observe that CKKS is more efficient than BFV (A scheme similar to BGV).

5 Conclusions (10 points)

Through the analysis and evaluation of CKKS and BGV, we have concluded that while very similar in structure and precision, each FHE scheme provides different benefits depending on certain scenarios. CKKS proved to be highly efficient in handling approximate arithmetic, and suitable for real-world numbers, making them extremely beneficial in machine learning applications. BGV was developed as a more general FHE scheme, with the ability to handle both integer and polynomial numbers. Both achieve high levels of security from their reliance on the RLWE problem. BGV and CKKS provide users with parameter customization, it would be interesting to compare the parameter selection process between both the schemes. This would help users take into account choosing parameters for specific applications of either scheme, expanding on the comparison between them as well. Security parameters for BGV and CKKS usually range from 128 to 256 bits, giving researchers a good starting point.

References

[1] J.-P. Bossuat, C. Mouchet, J. Troncoso-Pastoriza, and J.-P. Hubaux. Efficient bootstrapping for approximate homomorphic encryption with non-sparse keys. In A. Canteaut and F.-X. Standaert, editors,

Advances in Cryptology – EUROCRYPT 2021, pages 587–617, Cham, 2021. Springer International Publishing.

[2] H. Chen and K. Han. Homomorphic lower digits removal and improved fhe bootstrapping. In J. B. Nielsen and V. Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018*, pages 315–337, Cham, 2018. Springer International Publishing.

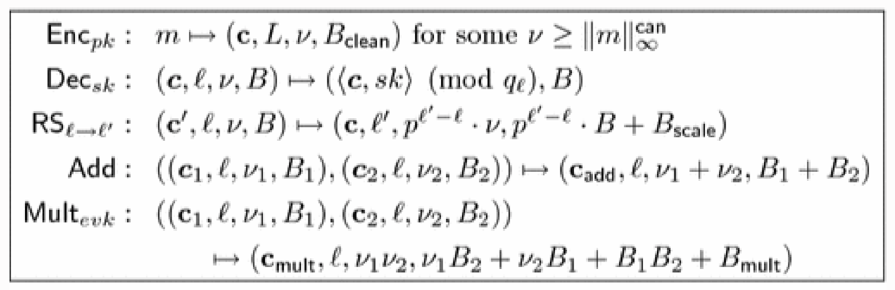
[3] J. H. Cheon, A. Kim, M. Kim, and Y. Song. Homomorphic encryption for arithmetic of approximate numbers. In T. Takagi and T. Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017*, pages 409–437, Cham, 2017. Springer International Publishing.

[4] S. Halevi and V. Shoup. Faster homomorphic linear transformations in helib. In H. Shacham and A. Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018*, pages 93–120, Cham, 2018. Springer International Publishing.

[5] R. Koseki, A. Ito, R. Ueno, M. Tibouchi, and N. Homma. Homomorphic encryption for stochastic computing. *J. Cryptogr. Eng.*, 13(2):251–263, June 2023.

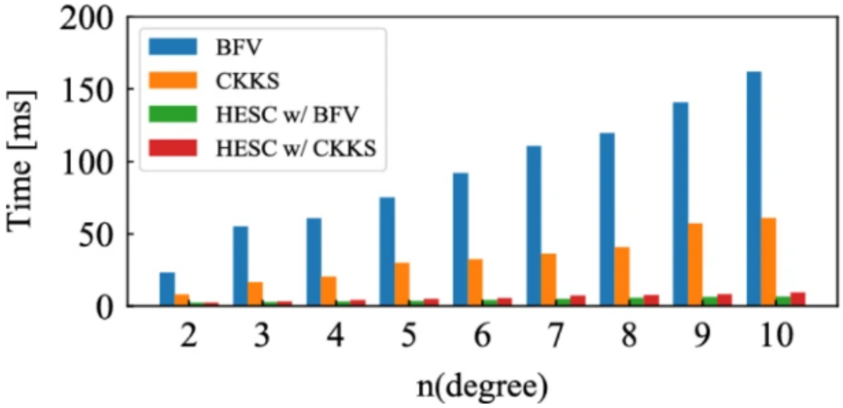
[6] B. Li and D. Micciancio. On the security of homomorphic encryption on approximate numbers. In A. Canteaut and F.-X. Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021*, pages 648–677, Cham, 2021. Springer International Publishing.

A Appendix



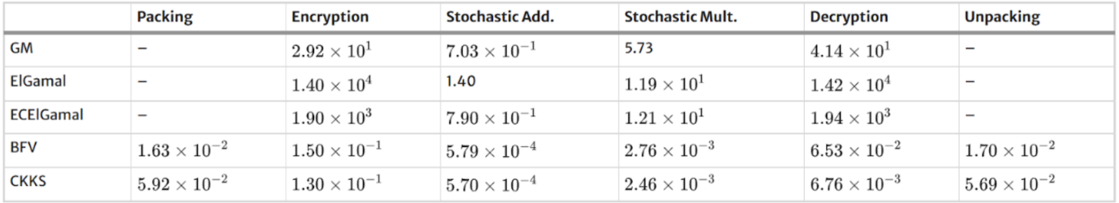
$$\begin{aligned}
\mathsf{Enc}_{pk}:&\quad m \mapsto (\mathbf{c}, L, \nu, B_{\mathsf{clean}}) \text{ for some } \nu \geq \|m\|_\infty^{\mathsf{can}}\\
\mathsf{Dec}_{sk}:&\quad (\mathbf{c}, \ell, \nu, B) \mapsto (\langle \mathbf{c}, sk \rangle \pmod{q_\ell}, B)\\
\mathsf{RS}_{\ell \to \ell'}:&\quad (\mathbf{c}', \ell, \nu, B) \mapsto (\mathbf{c}, \ell', p^{\ell'-\ell} \cdot \nu, p^{\ell'-\ell} \cdot B + B_{\mathsf{scale}})\\
\mathsf{Add}:&\quad ((\mathbf{c}_1, \ell, \nu_1, B_1), (\mathbf{c}_2, \ell, \nu_2, B_2)) \mapsto (\mathbf{c}_{\mathsf{add}}, \ell, \nu_1 + \nu_2, B_1 + B_2)\\
\mathsf{Mult}_{evk}:&\quad ((\mathbf{c}_1, \ell, \nu_1, B_1), (\mathbf{c}_2, \ell, \nu_2, B_2))\\
&\qquad \mapsto (\mathbf{c}_{\mathsf{mult}}, \ell, \nu_1\nu_2, \nu_1 B_2 + \nu_2 B_1 + B_1 B_2 + B_{\mathsf{mult}})
\end{aligned}$$

Figure 2: Description of CKKS [3]



Comparison of computation times in four schemes

Figure 3: BGV vs CKKS Cipher Text Structure [5]



	Packing	Encryption	Stochastic Add.	Stochastic Mult.	Decryption	Unpacking
GM	–	2.92×10^{1}	7.03×10^{-1}	5.73	4.14×10^{1}	–
ElGamal	–	1.40×10^{4}	1.40	1.19×10^{1}	1.42×10^{4}	–
ECElGamal	–	1.90×10^{3}	7.90×10^{-1}	1.21×10^{1}	1.94×10^{3}	–
BFV	1.63×10^{-2}	1.50×10^{-1}	5.79×10^{-4}	2.76×10^{-3}	6.53×10^{-2}	1.70×10^{-2}
CKKS	5.92×10^{-2}	1.30×10^{-1}	5.70×10^{-4}	2.46×10^{-3}	6.76×10^{-3}	5.69×10^{-2}

Figure 4: Description of CKKS [5]

B New Construction (if any, this is for 1-5 bonus points)

Similar to the conclusion, the new construction we suggest involves optimizing the homomorphic encryption parameters to enhance BGV and CKKS. Both schemes allow users to alter lattice dimensions, noise parameters, and the security variable. We are especially interested in working with the security parameter, as research shows stronger security leads to increased computational cost and communication overhead. While an increased security parameter strengthens security, it also lowers efficiency. An interesting construction to try would be increasing security parameters until the trade-off for efficiency and cost is optimized. Usual security parameters for BGV and CKKS are between 128 and 256 bits, giving a good initial start for testing.