On the overflow and p-adic theory applied to homomorphic encryption

Jacob Blindenbach[1,2,3], Jung Hee Cheon[4,5], Gamze Gürsoy[1,2,3], and Jiayi Kang[6] ⋆

[1] Department of Computer Science, Columbia University, USA
[2] Department of Biomedical Informatics, Columbia University, USA
[3] New York Genome Center, USA
jb4816@columbia.edu
gg2845@cumc.columbia.edu
[4] Seoul National University, Republic of Korea
[5] CryptoLab Inc., Republic of Korea
jhcheon@snu.ac.kr
[6] COSIC, KU Leuven, Belgium
jiayi.kang@esat.kuleuven.be

Keywords: Overflow, Homomorphic encryption, p-adic theory

Abstract. When integer and rational arithmetics are performed using modular arithmetics over $\mathbb{Z}/q\mathbb{Z}$, overflows naturally occur due to the mismatch between the infinite cardinality of $\mathbb{Z}$ or $\mathbb{Q}$ and the finite cardinality of $\mathbb{Z}/q\mathbb{Z}$. Since $\mathbb{Z}/q\mathbb{Z}$ is also the (sub) message space for many secure computation designs, secure computations of integer and rational arithmetics using these schemes must also consider the overflow problem. Previous works [CLPX, CT-RSA'18] and [HDRdS, ACNS'23] perform integer and rational arithmetics using the CLPX homomorphic encryption scheme, where overflows are avoided by restricting supported circuits. This introduces an additional constraint beyond the noise budget limitation. In our work, we discuss the possibilities of tolerating overflows. Firstly, we explain that when input messages and the final result are well-bounded, intermediate values can go arbitrarily large without affecting output correctness. This kind of overflow is called pseudo-overflow and does not need to be avoided. Secondly, we note that for prime-power modulus $q = p^r$, overflow errors are small in the p-adic norm. Therefore, we apply the p-adic encoding technique in [HDRdS, ACNS'23] to the BGV/BFV homomorphic encryption scheme with plaintext modulus p^r. Compared to [CLPX, CT-RSA'18] and [HDRdS, ACNS'23], our method supports circuits that are up to 2× deeper under the same ciphertext parameters, at the cost of an output error bounded by p^{-r} in the p-adic norm.

⋆ Work partially done while visiting Seoul National University

1 Introduction

The *overflow* phenomenon, also referred to as an overflow error, is a natural consequence of using a finite-storage machine for arithmetic over an infinite message space $(I, +, \times)$. Specifically, overflow occurs when an arithmetic operation in I leads to a numeric value that exceeds the predefined storage range.

The procedure of using the finite space $\mathbb{Z}/q\mathbb{Z}$ to perform integer and rational arithmetic is as follows:

1. $\mathsf{Encode} : I \longrightarrow \mathbb{Z}/q\mathbb{Z}$, where an integer message α is encoded into $\alpha \bmod q$ and a rational message $\frac{a}{b}$ where $gcd(a, b) = gcd(b, q) = 1$ is encoded into $a \cdot (b^{-1} \bmod q) \bmod q$.
2. Compute a function f composed of additions and multiplications in $\mathbb{Z}/q\mathbb{Z}$.
3. $\mathsf{Decode} : \mathbb{Z}/q\mathbb{Z} \longrightarrow I$.

In such settings, we distinguish two types of overflows: a *pseudo-overflow* [16] which corresponds to arbitrarily large intermediate values but a correct final result, and other overflows with incorrect outputs. In our work, the second type of overflow is referred to as a *persistent overflow*.

Overflow in Homomorphic Encryption. Homomorphic encryption (HE) is a cryptographic tool that allows computations over encrypted data without decrypting intermediate values. This feature is crucial for outsourcing computations involving sensitive information, such as genetic and financial data [22,26,4,8].

Since the finite space $\mathbb{Z}/q\mathbb{Z}$ is a (sub-)space for the BGV [6]/BFV [5,12] scheme and their variant CLPX [7], evaluating a function f on an input message $m \in I$ can be outsourced securely by following path 2 in Figure 1.

$$\begin{tikzcd}
\mathsf{Enc}\,(\mathsf{Encode}(m)) \arrow[r, "\mathsf{HE}\,.\,\mathsf{Eval}(f)"] & \mathsf{Enc}\,(f(\mathsf{Encode}(m))) \arrow[d, "\mathsf{Dec}"] \\
\mathsf{Encode}(m) \in \mathbb{Z}/q\mathbb{Z} \arrow[u, "\mathsf{Enc}"] \arrow[r, phantom, "\text{path 2}"'] & f(\mathsf{Encode}((m)) \in \mathbb{Z}/q\mathbb{Z} \arrow[d, "\mathsf{Decode}"] \\
m \in I \arrow[u, "\mathsf{Encode}"] \arrow[r, "f", "\text{path 1}"'] & f(m) \in I.
\end{tikzcd}$$

Fig. 1. Secure outsourcing of the computation f on the message space I, where Enc and Dec denote homomorphic encryption and decryption, and $\mathsf{HE}\,.\,\mathsf{Eval}(f)$ denotes the homomorphic evaluation of f in the ciphertext space.

Ciphertexts in HE contain *noise* components, which grows with homomorphic operations. The heuristic noise growth is included in Appendix A. In Figure 1, the Dec step is correct if the ciphertext $\mathsf{Enc}\,(f(\mathsf{Encode}(m)))$ remains a sufficient noise budget. For the *levelled* setting without bootstrapping, this sets the first constraint on the supported function f.

In works [7,17] that perform integer and rational arithmetics using CLPX, another constraint is set on f to ensure the correctness of Decode by avoiding overflows. As will be shown in Table 1, this overflow constraint is mostly more restrictive than the noise constraint, further lowering the maximum multiplicative depth in supported functions.

By contrast, this paper discusses the possibilities of tolerating overflows. Firstly, for applications with sufficiently bounded inputs and outputs, only pseudo-overflows may occur. Since a pseudo-overflow does not influence the correctness of Decode, supported functions f do not need overflow restrictions.

Secondly, we observe that for prime-power modulus $q = p^r$, the persistent overflow error is small (bounded by p^{-r}) in the p-adic norm. Therefore, for applications with p-adic precision r, overflow errors are negligibly small hence can be tolerated. Different from the Euclidean norm, p-adic norms are non-Archimedean and exhibit a hierarchical structure [23,1,11], as detailed in Appendix B. This leads to emerging applications in various areas including theoretical physics [3,18], genetic code translation [10,21,3,11] and cognitive science [1,2,19,20,3]. For those that involve sensitive data, HE with sub-plaintext space $\mathbb{Z}/p^r\mathbb{Z}$ provides a promising solution for their secure outsourcing.

Notation. Let $\mathbb{Z}/q\mathbb{Z}$ denote the set of integers modulo q, where $[-\frac{q}{2}, \frac{q}{2})$ is the representative interval. The notation $\mathbb{Z}_p$ stands for the set of p-adic integers (*i.e.* integers and rational numbers with non-negative p-adic valuations, detailed explanations are in Appendix B).

2 The overflow in modular arithmetic

This section discusses performing arithmetics in the message space I using the finite space $\mathbb{Z}/q\mathbb{Z}$, where I can be the set of integers, rational numbers, or p-adic integers. Specifically, when evaluating function f on $m \in I$, the relation

$$\mathsf{Decode} \circ f \circ \mathsf{Encode}(m) = f(m)$$

holds as long as the input message m and the final computation result $f(m)$ are in a certain subset of the message space I. Otherwise, a persistent overflow happens, where the error is large in the absolute norm but small in the p-adic norm when $q = p^r$.

2.1 Integer arithmetic

When the message space I is the set of integers $\mathbb{Z}$, an element $\alpha \in \mathbb{Z}$ is encoded into $\mathbb{Z}/q\mathbb{Z}$ as $(\alpha \bmod q)$, and the decoding reinterprets an element in $\mathbb{Z}/q\mathbb{Z}$ into $\mathbb{Z}$. As such, for messages m bounded by $\frac{q}{2}$, evaluating f in the modular arithmetic returns a number $h \in (-\frac{q}{2}, \frac{q}{2}] \cap \mathbb{Z}$.

While the relation $h = f(m) \bmod q$ always holds, their absolute value are equal if and only if $f(m) \in (-\frac{q}{2}, \frac{q}{2}]$. Specifically, intermediate values can go beyond $(-\frac{q}{2}, \frac{q}{2}]$ and become even arbitrarily large, which is referred to as the pseudo-overflow and does not affect the correctness in decoding the final result.

However, if $f(m) \notin (-\frac{q}{2}, \frac{q}{2}]$, then a persistent overflow happens. As illustrated by the dimension-1 lattice in Figure 2, the error $|h - f(m)|$ is multiples of q.

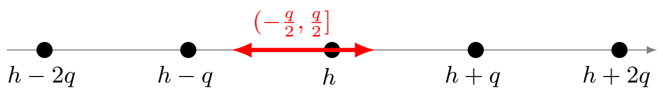

Fig. 2. Elements in $\mathbb{Z}$ that encodes to h form a one-dimensional lattice. In other words, for $h = \mathsf{Decode}(f(m))$, all the lattice elements are possible values of $f(m)$. If $f(m) \notin (-\frac{q}{2}, \frac{q}{2}]$, a persistent overflow occurs and $|\mathsf{Decode} \circ f \circ \mathsf{Encode}(m) - f(m)| = k \cdot q$ for some $k \geq 1$.

2.2 Rational arithmetic

Let $\mathbb{Q}^{(q)} = \{\frac{a}{b} \mid gcd(a,b) = gcd(b,q) = 1, b \neq 0\} \subset \mathbb{Q}$, then messages in $\mathbb{Q}^{(q)}$ can be encoded to $\mathbb{Z}/q\mathbb{Z}$ as follows [17]

$$\begin{aligned} \mathsf{Encode}\colon\ \mathbb{Q}^{(q)} &\longrightarrow \mathbb{Z}/q\mathbb{Z} \\ \frac{a}{b} &\to a \cdot (b^{-1} \bmod q) \bmod q. \end{aligned} \tag{1}$$

Since $\mathbb{Q}^{(q)}$ is an infinite set and $\mathbb{Z}/q\mathbb{Z}$ is only finite, the Encode map is not injective. Given $h = \mathsf{Encode}(\frac{a}{b}) \in \mathbb{Z}/q\mathbb{Z}$, the relation $a - bh = 0 \bmod q$ is satisfied. We observe that all the possible (a, b) that encode to the same element h are linear combinations of two independent solutions $(q, 0)$ and $(h, 1)$. In other words, these elements form a two-dimensional lattice $\mathcal{L}_h = \begin{bmatrix} q & 0 \\ h & 1 \end{bmatrix}$, as visualized in Figure 3.

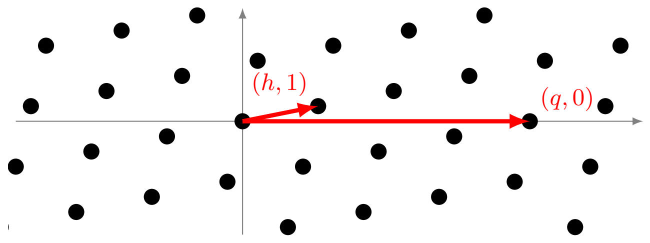

Fig. 3. Elements in $\mathbb{Q}^{(q)}$ that encodes to h forms a two-dimensional lattice $\mathcal{L}_h = \begin{bmatrix} q & 0 \\ h & 1 \end{bmatrix}$. If $f(m) \in \mathcal{F}_{N_q}$, then $\mathsf{Decode} \circ f \circ \mathsf{Encode}(m) = f(m)$. Otherwise, a persistent overflow occurs.

To get an invertible Encode map, the domain of (1) can be restricted into a subset $\mathcal{F}_{N_q} \subset \mathbb{Q}^{(q)}$, as suggested by existing works [16,25,17].

Lemma 1 (Restricting Encode to Farey rationals [16,25,17]). *Denote $N_q = \lfloor \sqrt{\frac{q-1}{2}} \rfloor$. Let $\mathcal{F}_{N_q} \subset \mathbb{Q}^{(q)}$ be Farey rationals of order N_q, i.e.*

$$\mathcal{F}_{N_q} = \left\{ \frac{a}{b} \mid \gcd(a,b) = \gcd(b,q) = 1, |a| \leq N_q, |b| \leq N_q \right\}.$$

Then $\mathsf{Encode}\mid_{\mathcal{F}_{N_q}}$ *is an injective map.*

We use the same encoding map (1) in $\mathcal{F}_{N_q}$, and we propose another decoding map that is different from the MEEA method in [25,17]. Precisely, we use the shortest vector in the lattice interpretation

$$\begin{aligned}\mathsf{Decode:}\ \mathbb{Z}/q\mathbb{Z} &\longrightarrow \mathcal{F}_{N_q}\\ h &\to \frac{a}{b} \text{ where } (a,b) = SVP(\mathcal{L}_h),\end{aligned}$$

where $SVP(\cdot)$ returns the shortest vector of a given lattice. It is easy to see the asymptotic relation $\mathcal{O}(SVP(\mathcal{L}_h)) = \mathcal{O}(\mathcal{F}_{N_q}) = \mathcal{O}(\sqrt{q})$ holds. Therefore, suppose input messages $m_1, \ldots, m_k \in \mathcal{F}_{N_q}$ are encoded to $\mathbb{Z}/q\mathbb{Z}$, and evaluating f in the modular arithmetic returns an element $h \in \mathbb{Z}/q\mathbb{Z}$. Then h will decode to $f(m_1, \ldots, m_k)$ correctly if and only if $f(m_1, \ldots, m_k)$ corresponds to the shortest vector in $\mathcal{L}_h$ in Figure 3.

For the k-variate polynomial f, let d be its total degree and t be its $L1$ norm. Then to ensure $f(m_1, \ldots, m_k) \in \mathcal{F}_{N_q}$, the work [17] restricts the input message space into $\mathcal{F}_M$, where $M \le \left(\frac{N_q}{t}\right)^{\frac{1}{dt}}$, *i.e.* $m_i \in \mathcal{F}_M \subset \mathcal{F}_{N_q}$. As a remark, there may exist applications whose outputs are always bounded for all possible inputs, i.e. $f(m_1, \ldots, m_k) \in \mathcal{F}_{N_q}, \forall m_i \in \mathcal{F}_{N_q}$. In such cases, there is no need to restrict the message space from $\mathcal{F}_{N_q}$ to $\mathcal{F}_M$ since only pseudo-overflows can occur.

Example 1 (Encode and Decode). In the example 1 of [17], $q = 3^{10}$ is used to encode rationals $a = 12.37 = \frac{1237}{100}$ and $b = 8.3 = \frac{83}{10}$. We correct erroneous results in [17] as follows. Following Lemma 1, we obtain $N_{3^{10}} = 171$ (not 125261 in [17]). Therefore, $a \notin \mathcal{F}_{N_{3^{10}}}$ and $b \in \mathcal{F}_{N_{3^{10}}}$, and as we will show, $\mathsf{Decode}(\mathsf{Encode}(a)) \neq a$ and $\mathsf{Decode}(\mathsf{Encode}(b)) = b$.

$$h_a = \mathsf{Encode}(\frac{1237}{100}) = 51385,\ \ h_b = \mathsf{Encode}(\frac{83}{10}) = 17723 \text{ (not 2196674185 and 9414317891 in [17])},$$

Using either MEEA or our lattice method to decode gives the same results:

$$\mathsf{Decode}(h_a) = -\frac{151}{131} \neq a,\ \mathsf{Decode}(h_b) = \frac{83}{10} = b.$$

Example 2 (Pseudo-overflow). Following the setting in Example 1 and let $c = 17 \in \mathcal{F}_{N_{3^{10}}}$, consider the evaluation of $f(b, c) = b + c - 16$.

$$\left(\mathsf{Encode}(\frac{83}{10}) + \mathsf{Encode}(17) - \mathsf{Encode}(16)\right) \bmod 3^{10} = (17723 + 17 - 16) \bmod 3^{10} = 17724$$

$\mathsf{Decode}(17724) = \dfrac{93}{10}$ gives the correct answer.

Note there exists pseudo-overflow during the computation

$$\mathsf{Decode}\left((\mathsf{Encode}(b) + \mathsf{Encode}(c)) \bmod 3^{10}\right) = \mathsf{Decode}(17740) = -\frac{10}{233} \neq \frac{253}{10},$$

but this does not influence the correctness of the final output in $f(b, c)$ since $f(b, c) \in \mathcal{F}_{N_{3^{10}}}$.

2.3 The p-adic arithmetic

In this subsection, we consider the modulus $q = p^r$ where p is a prime. Therefore, the subset $\mathbb{Q}^{(p^r)}$ is just the set of p-adic integers $\mathbb{Z}_p$. The previous Encode map (Equation 1) can then be re-interpreted as

$$\begin{aligned} \mathsf{Encode}\colon\ \mathbb{Z}_p &\longrightarrow \mathbb{Z}/p^r\mathbb{Z} \\ \alpha &\to H(p,r,\alpha) \bmod p^r, \end{aligned} \tag{2}$$

where $H(p,r,\alpha)$ is the Hensel code for a p-adic integer α with precision r. The definition of Hensel code is included in the Appendix B.2.

As such, the error between an infinite-length p-adic integer and its length-r encoding is bounded by p^{-r} in the p-adic norm. Due to the strong triangle inequality property of the p-adic norm (detailed in the Appendix B.1), performing arithmetics over $\mathbb{Z}/p^r\mathbb{Z}$ does not increase the error. In other words, $f(m)$ does not need to be an element in $\mathcal{F}_{N_{p^r}}$ and the p-adic norm of the persistent overflow error $(\mathsf{Decode} \circ f \circ \mathsf{Encode}(m) - f(m))$ is bounded by p^{-r}.

Example 3. Using the same parameters as in Example 1, we now consider $f'(b,c) = b + c$, and we have shown $\mathsf{Decode} \circ f' \circ \mathsf{Encode}(b,c) = -\frac{10}{233} \neq f'(b,c) = \frac{253}{10}$. Their 3-adic representations are $(\frac{253}{10})_3 = .1000010220022\cdots$ and $(-\frac{10}{233})_3 = .1000010220120\cdots$, hence $|f'(b,c) - \mathsf{Decode} \circ f' \circ \mathsf{Encode}(b,c)|_3 = 3^{-10}$, verifying that the overflow error is 3-adically small.

This observation provides another taste of parameter selections for applications that are endowed with p-adic norm: instead of restricting $f(m)$ to $\mathcal{F}_{N_{p^r}}$ to get error-free computations, it is also possible to free the overflow restriction at the cost of a p-adically small error.

3 Outsourcing computations endowed with the p-adic norm in Homomorphic Encryption

In this section, we propose to outsource applications endowed with a p-adic norm using $\mathsf{BGV}\,/\,\mathsf{BFV}$ with plaintext modulus p^r, where r is the desired precision in the p-adic norm. Let R_{p^r} denote the BGV plaintext space with modulus p^r and cyclotomic polynomial degree $\mathfrak{n}$, and ℓ denote the number of SIMD slots in R_{p^r}. Let $Q \gg p^r$ denote the ciphertext modulus. Then ℓ rational numbers and integers that are p-adic integers are encoded as follows

$$\begin{aligned} \mathsf{BGV.Encode}\colon\ \mathbb{Z}_p^\ell &\longrightarrow (\mathbb{Z}/p^r\mathbb{Z})^\ell \longrightarrow R_{p^r} \\ \prod_{i=1}^{\ell} \alpha_i &\to \prod_{i=1}^{\ell} h_i \to \mathsf{CRT}^{-1}(b_i), \end{aligned} \tag{3}$$

where $h_i = H(p,r,\alpha_i) \bmod p^r$ is the Hensel code of α_i with precision r and CRT^{-1} is the SIMD encoding map detailed in Appendix A. Its inverse procedure is

$$\begin{aligned} \mathsf{BGV.Decode}\colon R_{p^r} &\longrightarrow (\mathbb{Z}/p^r\mathbb{Z})^\ell \longrightarrow \mathbb{Z}_p^\ell \\ v &\to \mathsf{CRT}(v) = \prod_{i=1}^{\ell} h_i \to \prod_{i=1}^{\ell} \frac{a_i}{b_i}, \end{aligned} \tag{4}$$

where CRT is the SIMD decoding map detailed in Appendix A and $(a_i, b_i) = SVP(\mathcal{L}_{h_i})$.

For any input in a BGV slot $\alpha_i \in \mathbb{Z}_p$, the error e in a persistent overflow (*i.e.* the difference between the output following path 2 and path 1 in Figure 1) satisfies $|e|_p \leq p^{-r}$ and is negligible for the p-adic precision r. Therefore, supported functions in a homomorphic evaluation are not constrained by the overflow, but only noise budgets. This is different from previous designs [7,17] where both constraints are considered.

To ensure a fair comparison between our method (p-adic encoding into BGV with plaintext modulus p^r) and previous works [7,17], we consider schemes with the same ciphertext parameters (thus having the same costs for homomorphic additions and multiplications) when evaluating parameterized circuits. Specifically, we consider a regular circuit [9] that consists of D multiplicative levels with at most A levels of additions in each multiplicative level. Then for fixed $A = 0, 3$, we compare the maximum multiplicative depth D in supported circuits for both methods. The comparison result is presented in Table 1, where parameters $(\mathfrak{n}, \log_2 Q, b, L)$ are taken from the Table 3 in [17].

As shown in Table 1, for the parameter $(\mathfrak{n} = 2^{15}, \log_2 Q = 890, L = 2^8)$, our method supports circuits that are twice deeper than [17,7]. In general, the maximum multiplicative depth of supported circuits in our approach is always never lower than that of [17], and also greater than that of [7] for most parameters. Beside this depth advantage, our approach is compatible with bootstrapping and the optimized polynomial evaluation strategy [28], as detailed in Appendix C.

We also implement $\mathsf{BGV.Encode}$ and $\mathsf{BGV.Decode}$ as a wrapper to the `HElib` library v2.3.0[7], which is available in https://github.com/G2Lab/padicBGV. The repository also contains detailed documentation and instructions for usage.

4 Conclusion

In this work, we explained the overflow in modular arithmetic in $\mathbb{Z}/q\mathbb{Z}$ using a heuristic lattice interpretation, distinguishing pseudo-overflows from persistent overflows and correcting errors in the examples of [17].

Moreover, for applications that involve sensitive data and desire a p-adic precision r, we propose to use $\mathsf{BGV}/\mathsf{BFV}$ with plaintext modulus p^r for its secure computation. Compared with previous works [7,17], our approach supports circuits that are up to $2\times$ deeper under the same ciphertext parameters, and it is also compatible with bootstrapping. For future works, further investigations of p-adic applications with privacy concerns would be valuable to apply our method.

[7] https://github.com/homenc/HElib

Number of additions $A = 0$														
		$L = 2^8$						$L = 2^{16}$						
$\mathfrak{n}$	$\log_2 Q$	b	t	D_n	D_o	D	$\lvert e\rvert_2$	b	t	D_n	D_o	D	$\lvert e\rvert_2$	Method
2^{14}	435	257	—	15	14	14	0	257	—	15	13	13	0	[7]
		2^{16}	—	11	11	11	0	2^{16}	—	11	11	11	0	[17]
		—	2^8	15	—	**15**	2^{-8}	—	2^{16}	11	—	**11**	2^{-16}	Ours
2^{15}	890	2^{16}	—	23	16	16	0	2^{16}	—	23	15	15	0	[7]
		2^{16}	—	23	15	15	0	2^{16}	—	23	14	14	0	[17]
		—	2^8	32	—	**32**	2^{-8}	—	2^{16}	24	—	**24**	2^{-16}	Ours
Number of additions $A = 3$														
2^{14}	435	128	—	14	13	13	0	2^{11}	—	12	13	12	0	[7]
		2^{16}	—	10	10	10	0	2^{16}	—	10	10	10	0	[17]
		—	2^8	14	—	**14**	2^{-8}	—	2^{16}	10	—	**10**	2^{-16}	Ours
2^{15}	890	2^{28}	—	16	16	16	0	2^{22}	—	18	15	15	0	[7]
		2^{16}	—	22	15	15	0	2^{16}	—	22	14	14	0	[17]
		—	2^8	28	—	**28**	2^{-8}	—	2^{16}	22	—	**22**	2^{-16}	Ours

Table 1. Comparison of the maximum multiplicative depth D of supported circuits in [7], [17] and our p-adic encoding to BGV for different input size L and regular circuits [9] parameterized by A. Letters D_n and D_o denote the depth bound from noise and overflow respectively, which are estimated following the heuristic analysis in [7] and in Appendix A. The maximum depth D is supported circuits equals $\min\{D_n, D_o\}$ in [7], [17] and D_n in our method. The notation $|e|_2$ denotes the maximum output error in the 2-adic norm.

Acknowledgements. This work was supported by CyberSecurity Research Flanders with reference number VR20192203, Industrial & Mathematical Data Analytics Research Center (2022R1A5A6000840), and the National Science Foundation Graduate Research Fellowship under Grant No. (DGE-2036197).

The authors would also like to thank Frederik Vercauteren for giving feedback on this paper, and thank Jai Hyun Park, Hyeongmin Choe for valuable discussions.

References

1. Albeverio, S., Khrennikov, A., Tirozzi, B., De Smedt, S.: p-adic dynamic systems. Theoretical and mathematical physics **114**, 276–287 (1998)
2. Albeverio, S., Kloeden, P., Khrennikov, A.: Human memory as a p-adic dynamic system. Theoretical and mathematical physics **117**(3), 1414–1422 (1998)
3. Anashin, V., Khrennikov, A.: Applied algebraic dynamics. Walter de Gruyter (2009)
4. Blindenbach, J., Kang, J., Hong, S., Karam, C., Lehner, T., Gürsoy, G.: Ultra-secure storage and analysis of genetic data for the advancement of precision medicine. bioRxiv (2024). https://doi.org/10.1101/2024.04.16.589793
5. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical gapsvp. In: Safavi-Naini, R., Canetti, R. (eds.) Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA,

August 19-23, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7417, pp. 868–886. Springer (2012). https://doi.org/10.1007/978-3-642-32009-5_50

6. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. In: Goldwasser, S. (ed.) Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012. pp. 309–325. ACM (2012). https://doi.org/10.1145/2090236.2090262
7. Chen, H., Laine, K., Player, R., Xia, Y.: High-precision arithmetic in homomorphic encryption. In: Smart, N.P. (ed.) Topics in Cryptology - CT-RSA 2018 - The Cryptographers' Track at the RSA Conference 2018, San Francisco, CA, USA, April 16-20, 2018, Proceedings. Lecture Notes in Computer Science, vol. 10808, pp. 116–136. Springer (2018). https://doi.org/10.1007/978-3-319-76953-0_7
8. Cong, K., Kang, J., Nicolas, G., Park, J.: Faster private decision tree evaluation for batched input from homomorphic encryption. Cryptology ePrint Archive, Paper 2024/662 (2024), https://eprint.iacr.org/2024/662
9. Costache, A., Smart, N.P., Vivek, S., Waller, A.: Fixed-point arithmetic in SHE schemes. In: Avanzi, R., Heys, H.M. (eds.) Selected Areas in Cryptography - SAC 2016 - 23rd International Conference, St. John's, NL, Canada, August 10-12, 2016, Revised Selected Papers. Lecture Notes in Computer Science, vol. 10532, pp. 401–422. Springer (2016). https://doi.org/10.1007/978-3-319-69453-5_22
10. Dragovich, B., Dragovich, A.Y.: A p-adic model of dna sequence and genetic code. P-Adic Numbers, Ultrametric Analysis, and Applications **1**(1), 34–41 (Feb 2009). https://doi.org/10.1134/s2070046609010038
11. Dragovich, B., Misic, N.Z.: p-adic hierarchical properties of the genetic code. Biosystems **185**, 104017 (2019). https://doi.org/10.1016/j.biosystems.2019.104017
12. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. IACR Cryptol. ePrint Arch. p. 144 (2012), http://eprint.iacr.org/2012/144
13. Geelen, R., Beirendonck, M.V., Pereira, H.V.L., Huffman, B., McAuley, T., Selfridge, B., Wagner, D., Dimou, G.D., Verbauwhede, I., Vercauteren, F., Archer, D.W.: BASALISC: programmable hardware accelerator for BGV fully homomorphic encryption. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2023**(4), 32–57 (2023). https://doi.org/10.46586/TCHES.V2023.I4.32-57
14. Geelen, R., Iliashenko, I., Kang, J., Vercauteren, F.: On polynomial functions modulo p^{e} and faster bootstrapping for homomorphic encryption. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part III. Lecture Notes in Computer Science, vol. 14006, pp. 257–286. Springer (2023). https://doi.org/10.1007/978-3-031-30620-4_9
15. Gouvêa, F.: p-adic Numbers: An Introduction. Universitext, Springer International Publishing (2020)
16. Gregory, R.: Error-free computation with rational numbers. BIT Numerical Mathematics **21**(2), 194–202 (1981)
17. Harmon, L., Delavignette, G., Roy, A., da Silva, D.W.H.A.: PIE: p-adic encoding for high-precision arithmetic in homomorphic encryption. In: Tibouchi, M., Wang, X. (eds.) Applied Cryptography and Network Security - 21st International Conference, ACNS 2023, Kyoto, Japan, June 19-22, 2023, Proceedings, Part I. Lecture Notes in Computer Science, vol. 13905, pp. 425–450. Springer (2023). https://doi.org/10.1007/978-3-031-33488-7_16
18. Khrennikov, A.Y., Oleschko, K., López, M.d.J.C.: Applications of p-adic numbers: from physics to geology. Contemp. Math. **665**, 121–131 (2016)

19. Khrennikov, A.: Human subconscious as a p-adic dynamical system. Journal of Theoretical Biology **193**(2), 179–196 (1998). https://doi.org/10.1006/jtbi.1997.0604
20. Khrennikov, A.Y., Nilsson, M.: P-adic deterministic and random dynamics, vol. 574. Springer Science & Business Media (2004)
21. Khrennikov, A.Y.: Gene expression from polynomial dynamics in the 2-adic information space. Chaos, Solitons & Fractals **42**(1), 341–347 (2009)
22. Kim, M., Harmanci, A.O., Bossuat, J.P., Carpov, S., Cheon, J.H., Chillotti, I., Cho, W., Froelicher, D., Gama, N., Georgieva, M., Hong, S., Hubaux, J.P., Kim, D., Lauter, K., Ma, Y., Ohno-Machado, L., Sofia, H., Son, Y., Song, Y., Troncoso-Pastoriza, J., Jiang, X.: Ultrafast homomorphic encryption models enable secure outsourcing of genotype imputation. Cell Systems **12**(11), 1108–1120.e4 (2021). https://doi.org/10.1016/j.cels.2021.07.010
23. Koblitz, N.: p-adic Numbers, p-adic Analysis, and Zeta-Functions. Springer New York, NY, 2 edn. (1984)
24. Koc, C.K.: A tutorial on p-adic arithmetic. Tech. rep., Oregon State University (2002)
25. Kornerup, P., Gregory, R.T.: Mapping integers and hensel codes onto farey fractions. BIT Numerical Mathematics **23**, 9–20 (1983)
26. Li, W., Kim, M., Zhang, K., Chen, H., Jiang, X., Harmanci, A.: Collagene enables privacy-aware federated and collaborative genomic data analysis. Genome Biology **24**(1), 204 (2023)
27. Ma, S., Huang, T., Wang, A., Wang, X.: Accelerating bgv bootstrapping for large p using null polynomials over $\mathbb{Z}_{p^e}$. Cryptology ePrint Archive, Paper 2024/115 (2024), https://eprint.iacr.org/2024/115
28. Okada, H., Player, R., Pohmann, S.: Homomorphic polynomial evaluation using galois structure and applications to BFV bootstrapping. IACR Cryptol. ePrint Arch. p. 1304 (2023), https://eprint.iacr.org/2023/1304
29. Paterson, M., Stockmeyer, L.J.: On the number of nonscalar multiplications necessary to evaluate polynomials. SIAM J. Comput. **2**(1), 60–66 (1973). https://doi.org/10.1137/0202007
30. Smart, N.P., Vercauteren, F.: Fully homomorphic SIMD operations. Des. Codes Cryptogr. **71**(1), 57–81 (2014). https://doi.org/10.1007/S10623-012-9720-4

A Introduction to the BGV/BFV and the CLPX scheme

Notation. Let $\Phi_{\mathfrak{m}}(X)$ denote the $\mathfrak{m}$-th cyclotomic polynomial with degree $\mathfrak{n} = \phi(\mathfrak{m})$, where $\phi(\cdot)$ is the Euler totient function. Let $R = \mathbb{Z}[X]/\Phi_{\mathfrak{m}}(X)$, $R_t = R/tR$ and $R_Q = R/QR$.

Basic setup of BGV/BFV. In the BGV/BFV scheme, the plaintext space is R_t and the ciphertext space is R_Q^2. Basic setups include key generation, encryption and decryption.

– $\mathsf{KeyGen}(\mathsf{params})$: for given parameters, generate the secret key sk, public key pk, and key switching keys for relinearisation rlk and automorphisms aut.
– $\mathsf{Enc}_{\mathsf{pk}}(\mathsf{pt})$: using the public key pk, the algorithm outputs a ciphertext $\mathsf{ct} \in R_Q^2$ that encrypts $\mathsf{pt} \in R_t$.

– $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct})$: using the secret key sk, the algorithm outputs a plaintext $\mathsf{pt} \in R_t$ that corresponds to $\mathsf{ct} \in R_Q^2$. The decryption is correct if the noise component in ct is properly bounded.

Homomorphic operations in BGV/BFV. Let ct_i denote ciphertexts that encrypt pt_i.

– $\mathsf{HE}\,.\,\mathsf{Add}(\mathsf{ct}_0, \mathsf{ct}_1)$: the algorithm outputs a ciphertext that encrypts $\mathsf{pt}_0 + \mathsf{pt}_1 \in R_t$.
– $\mathsf{HE}\,.\,\mathsf{Mult}(\mathsf{ct}_0, \mathsf{ct}_1, \mathsf{rlk})$: the algorithm outputs a ciphertext that encrypts $\mathsf{pt}_0 \cdot \mathsf{pt}_1 \in R_t$.
– $\mathsf{HE}\,.\,\mathsf{Aut}(\mathsf{ct}, \mathsf{aut})$: the algorithm outputs a ciphertext that encrypts the result of applying an automorphism to pt.

It is worth noting that the HE . Mult and HE . Aut both include a subprocedure KeySwitch, which is up to $\times 145$ times [13] more expensive than HE . Add. Therefore, the running time of a homomorphic evaluation is mainly determined by the number of KeySwitchs.

In the levelled version of the BGV/BFV scheme, its noise capacity τ is predetermined by parameters $(Q, t, \mathfrak{n})$. Therefore, circuits $\mathcal{C}$ that consume lower noise budgets than τ can be correctly evaluated. As noise consumption is mainly determined by the number of consecutive homomorphic multiplications, the multiplicative depth of supported circuits in $\mathcal{C}$ are upper-bounded.

Meanwhile, a levelled BGV/BFV can be turned into a fully homomorphic encryption (FHE) scheme using bootstrapping. Bootstrapping is an operation for noise refreshing, which allows the evaluation of arbitrary circuits. While a KeySwitch operation takes less than 1 second [13], a bootstrapping requires could take more than 14 minutes [14,27]. Therefore, accelerating BGV/BFV bootstrapping is an interesting and important research topic, yet most existing privacy-preserving applications use levelled BGV/BFV to avoid bootstrapping.

SIMD packing in BGV/BFV *[30].* Consider the plaintext modulus $t = p^r$ for some prime p. If p is coprime to $\mathfrak{m}$, then the polynomial $\Phi_{\mathfrak{m}}(X)$ splits modulo p^r into ℓ irreducible factors of same degree d

$$\Phi_{\mathfrak{m}}(X) = F_1(X) \cdots F_\ell(X),$$

where d is the order of p modulo $\mathfrak{m}$, and $\ell = \mathfrak{n}/d$. Following the Chinese Reminder Theorem (CRT), the map

$$\mathsf{CRT} : R_{p^r} \to \prod_{i=1}^{\ell} \mathbb{Z}[X]/\big(p^r\,\mathbb{Z}, F_i(X)\big),$$

is an isomorphism with inverse CRT^{-1}. This therefore enables the encoding of ℓ messages $\{w_1 \ldots, w_\ell\} \in \prod_{i=1}^{\ell} \mathbb{Z}[X]/\big(p^r\,\mathbb{Z}, F_i(X)\big)$ into a single plaintext in R_{p^r} and their procession in the SIMD (Single-Instruction Multiple-Data) manner.

CLPX *as a variant of* BFV. The CLPX scheme is a variant of the BFV scheme whose cyclotomic order $\mathfrak{m}$ is a power-of-2. In such cases, the cyclotomic polynomial $\Phi_{\mathfrak{m}}(X) = X^{\mathfrak{n}} + 1$.

The ciphertext space, KeyGen, $\mathsf{HE.Add}$, $\mathsf{HE.Mult}$ in CLPX are identical to the BFV scheme. However, the plaintext modulus t in BFV is replaced by a polynomial $(X - b)$. As such, the CLPX plaintext space is

$$R/(X-b) \cong \mathbb{Z}/(b^{\mathfrak{n}}+1)\,\mathbb{Z}\,.$$

With this adaption, $\mathsf{HE.Aut}$ is no longer supported, and the noise growth in homomorphic multiplications is (almost) proportional to the parameter b instead of to the plaintext modulus t. Precisely, let e_i denote the noise in ciphertext $\mathsf{ct}_i \in R_Q^2$ for $i = 1, 2$ and e_{mult} denote the noise in their homomorphic product, then

$$\|e_{mult}\|_{\mathsf{BGV\,/\,BFV}} \lesssim 14t\mathfrak{n}\ \max\{\|e_1\|, \|e_2\|\}$$
$$\|e_{mult}\|_{\mathsf{BGV\,/\,BFV}} \lesssim 14(b+1)\mathfrak{n}\ \max\{\|e_1\|, \|e_2\|\}$$

are satisfied, as explained in [7]. Therefore, if $b \ll t$, then CLPX supports modular arithmetics with a higher depth bound than $\mathsf{BGV/BFV}$ under the same parameters $(Q, \mathfrak{n})$. This is quantified in Lemma 2.

Lemma 2 (Noise bound [7]). *Consider a regular circuit that consists of D multiplicative levels with at most A levels of additions in each multiplicative level. Then the multiplicative depth of circuits whose outputs are correctly decrypted need to satisfy the following conditions:*

$$D_{\mathsf{BGV\,/\,BFV}} \lesssim \left\lfloor \frac{\log Q - \log(84\sigma t\mathfrak{n})}{\log(14tn) + A} \right\rfloor \tag{5}$$

$$D_{\mathsf{CLPX}} \lesssim \left\lfloor \frac{\log Q - \log(2(b+1)^2 n^{3/2})}{\log(14(b+1)n) + A} \right\rfloor \tag{6}$$

To date, there has been no known design for the CLPX bootstrapping. Therefore, the CLPX scheme can only be used in a levelled manner with constrained circuits. Moreover, in CLPX the SIMD packing is different from $\mathsf{BGV/BFV}$. Consider the prime factorization $b^{\mathfrak{n}} + 1 = \prod_{i=1}^{\ell'} p_i^{r_i}$ for prime p_i, the isomorphism

$$\mathbb{Z}/(b^{\mathfrak{n}}+1)\,\mathbb{Z} \cong \prod_{i=1}^{\ell'} \mathbb{Z}/p_i^{r_i}\,\mathbb{Z}$$

enables the encoding of ℓ' messages $\{w_1 \ldots, w_{\ell'}\} \in \prod_{i=1}^{\ell'} \mathbb{Z}/p_i^{r_i}\,\mathbb{Z}$ into a single CLPX plaintext. While slots in $\mathsf{BGV/BFV}$ have the same cardinality, in CLPX the slot sizes $p_i^{r_i}$ vary, potentially restricting its practical applicability.

B The p-adic norm and p-adic integers

B.1 p-adic norm on the rational number field $\mathbb{Q}$

Definition 1. *A norm on a field F is a function*

$$\|\ \| : F \longrightarrow \mathbb{R}_+$$

that satisfies the following conditions

1. $\|x\| = 0$ *if and only if* $x = 0$*,*

2. $\|x \cdot y\| = \|x\| \cdot \|y\|, \ \forall x, y \in F,$
3. $\|x + y\| \leq \|x\| + \|y\|, \ \forall x, y \in F.$

Regarding the rational number field $\mathbb{Q}$, Ostrowski's Theorem [23] states that every non-trivial norm on $\mathbb{Q}$ is equivalent to either the usual absolute value or the p-adic norm for some prime p. This section proceeds to introduce the p-adic norm.

Let $\nu_p(\cdot)$ denote the *p-adic valuation* on $\mathbb{Z}$ defined as

$$\nu_p(m) = \begin{cases} \max\{k \in \mathbb{N} : p^k \mid m\} & \text{if } m \neq 0, \\ +\infty & \text{if } m = 0. \end{cases}$$

It generalizes to the rational numbers as $\nu_p(m/n) = \nu_p(m) - \nu_p(n)$. Moreover, a rational number with a non-negative p-adic valuation is a *p-adic integer*.

Proposition 1 (p-adic norm [23,15]). *Define the map $| \ \ |_p$ on $\mathbb{Q}$ as follows:*

$$|x|_p = \begin{cases} p^{-\nu_p(x)} & \textit{if } x \neq 0, \\ 0 & \textit{if } x = 0. \end{cases}$$

Then $| \ \ |_p$ is a norm on $\mathbb{Q}$, and is called a p-adic norm.

While the usual absolute value $| \ \ |$ is *Archimedean*, i.e. given $x, y \in \mathbb{Q}$ and $x \neq 0$, there always exists $n \in \mathbb{N}^+$ such that $|nx| > |y|$, this does not hold for the p-adic norm. Specifically, the p-adic norm satisfies the *strong triangle inequality*

$$|a + b|_p \leq \max\{|a|_p, |b|_p\}, \forall a, b \in \mathbb{Q},$$

hence adding an element $x \in \mathbb{Q}$ to itself does not increase its p-adic norm. As such, the p-adic norm is also known to be *ultrametric*.

B.2 Representations of p-adic integers

Definition 2 (p-adic integer and representation [15,24]). *A p-adic integer $\alpha \in \mathbb{Z}_p$ is an infinite formal sum of the form*

$$\alpha = \alpha_0 + \alpha_1 p + \alpha_2 p^2 + \cdots + \alpha_i p^i + \cdots$$

where the digits $0 \leq \alpha_i \leq p - 1$ for all $i \geq 0$. Furthermore, it can be represented as

$$(\alpha)_p = .\alpha_0 \alpha_1 \alpha_2 \cdots \alpha_i \cdots$$

where . is called a p-adic point.

Property 1. If the p-adic valuation of α is i, then α_i is the first non-zero digit, and its p-adic norm $|\alpha|_p = p^{-i}$.

Property 2. The position of p-adic point is shifted by multiplying powers of p.

Moreover, the infinite-length representation can be approximated by a Hensel code.

Definition 3 (Hensel code [24]). *The Hensel code $H(p,r,\alpha)$ for a p-adic integer α is a length-r segment of its infinite p-adic expansion. Precisely, let $(\alpha)_p = .\alpha_0\alpha_1\alpha_2\cdots\alpha_i\cdots$, then*

$$H(p,r,\alpha) = .\alpha_0\alpha_1\cdots\alpha_{r-1}\ .$$

As such, the approximation error $|\alpha - H(p,r,\alpha)|_p = p^{-r}$ is small in the p-adic norm.

B.3 The hierarchical structure of p-adic numbers

The most distinguishing feature of the p-adic norm from the Euclidean norm is its hierarchical structure [10,19,18,3,20,1,21], sometimes also referred to as the fractal structure [11]. This property is visualized in Figure 4 and Figure 5 for $p = 3$.

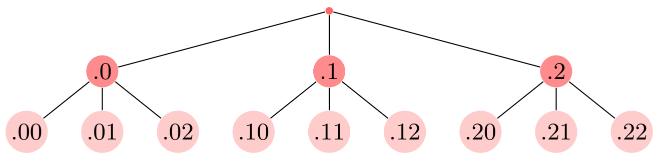

Fig. 4. Hierarchical structure for 3-adic numbers visualized as a tree. The 3-adic distance between two leaves is determined by the level of their common ancestor. For example, the element .01 is 3-adically closer to .02 than to .21 due to the common ancestor .0. This is verified by computing their p-adic distance: $|.01 - .02|_3 = 3^{-1}$,$|.01 - .21|_3 = 3^0 = 1$ and $3^{-1} < 1$.

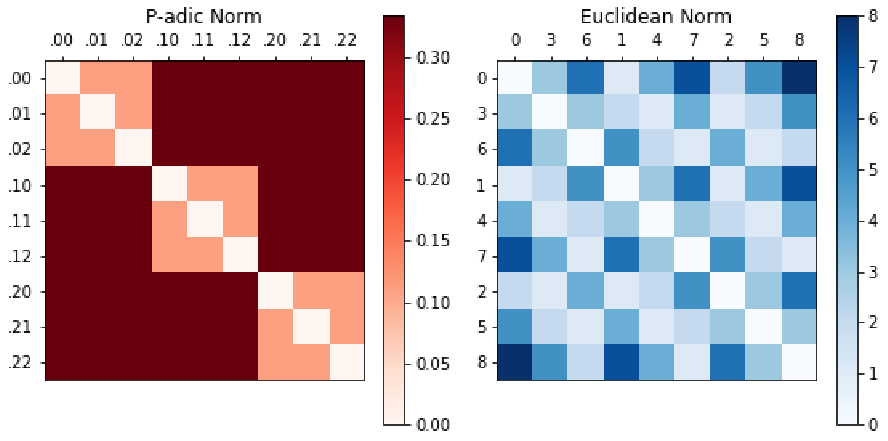

Fig. 5. Heatmap illustrating the relationship between the 3-adic norm and the Euclidean norm ($L2$-norm), where the 3-adic norm exhibits a fractal structure.

C Additional advantages of our method in Section 3

Combination with bootstrapping. To date, there exist bootstrapping designs for the BGV scheme, but not for the CLPX scheme. While Table 1 provides a comparison of the three designs using CLPX and BGV in a levelled manner, a natural question to ask is about potential changes in incorporating bootstrapping.

As bootstrapping allows noise refreshing, we take $D_n = \infty$. As such, for [7] and [17] the maximum depth is still constrained by overflow: $D = D_o$; for our method $D = D_n = \infty$, meaning evaluations of arbitrary circuits are supported.

Combination with optimzed polynomial evaluation strategy. Polynomials are commonly evaluated as sequences of HE . Adds and HE . Mults, and widely used methods (such as the Paterson-Stockmeyer method [29]) require $\mathcal{O}(\sqrt{d})$ HE . Mults when evaluating a degree-d polynomial. Nevertheless, a recent work [28] suggests to use HE . Aut for polynomial evaluations. which reduces the number of KeySwitchs in evaluating a degree-d polynomial into $\mathcal{O}(\log d)$. This optimization does not apply to [7,17], but it is possible to be combined with our approach, as demonstrated in Table 2.

p	d	ℓ	r	t	$\lvert e\rvert_p$	D	# $\mathsf{KeySwitch}_{\mathsf{Aut}}$	# $\mathsf{KeySwitch}_{PS}$
3	16384	2	3	3^3	3^{-3}	32	29	198
			6	3^6	3^{-6}	27		
5	16384	2	3	5^3	5^{-3}	29	29	198
			6	5^6	5^{-6}	23		
7	8192	4	3	7^3	7^{-3}	28	27	134
			6	7^6	7^{-6}	21		

Table 2. For $\mathfrak{n} = 2^{15}$ and $\log_2 q = 890$, the table presents the extension degree d, the number of SIMD slots $\ell = \mathfrak{n}/d$, and the number of KeySwitch operations in evaluating a degree-$(d+1)$ polynomial when using the norm-based approach in [28] (which includes HE . Aut and denoted as # $\mathsf{KeySwitch}_{\mathsf{Aut}}$) and using the Paterson-Stockmeyer method (# $\mathsf{KeySwitch}_{PS}$). The maximum multiplicative depth is estimated for regular circuits with 3 consecutive additions in each multiplicative level ($A = 3$). To correctly evaluate the degree-$(d+1)$ polynomial in the levelled BGV, the relation $D \geq \log_2(d+1)$ needs to be satisfied.