PIE: *p*-adic Encoding for High-Precision Arithmetic in Homomorphic Encryption

Luke Harmon[1], Gaetan Delavignette[1], Arnab Roy[1], and David Silva[1]

Algemetric
{lharmon,gdelavignette,dsilva}@algemetric.com
arnab.roy@windowslive.com

Abstract. A large part of current research in homomorphic encryption (HE) aims towards making HE practical for real-world applications. In any practical HE, an important issue is to convert the application data (type) to the data type suitable for the HE.

The main purpose of this work is to investigate an efficient HE-compatible encoding method that is generic, and can be easily adapted to apply to the HE schemes over integers or polynomials.

p-adic number theory provides a way to transform rationals to integers, which makes it a natural candidate for encoding rationals. Although one may use naive number-theoretic techniques to perform rational-to-integer transformations without reference to p-adic numbers, we contend that the theory of p-adic numbers is the proper lens to view such transformations.

In this work we identify mathematical techniques (supported by p-adic number theory) as appropriate tools to construct a generic rational encoder which is compatible with HE. Based on these techniques, we propose a new encoding scheme PIE that can be easily combined with both AGCD-based and RLWE-based HE to perform high precision arithmetic. After presenting an abstract version of PIE, we show how it can be attached to two well-known HE schemes: the AGCD-based IDGHV scheme and the RLWE-based (modified) Fan-Vercauteren scheme. We also discuss the advantages of our encoding scheme in comparison with previous works.

1 Introduction

A large part of current research and development in HE is focused on efficient implementation with suitable software and/or hardware support and developing practically usable libraries for HE that can be used for various machine learning and data analysis applications. These works clearly aim towards making HE practical for real-world applications.

The state-of-the-art HE schemes are defined to process (modulo) integer inputs or polynomial inputs (with modulo integer coefficients). For a significantly large number of practical applications, an HE scheme should be able to operate on real/rational numbers. In any practical HE an important issue is to convert the application data (type) to the data type suitable for the HE. This is usually achieved by encoding real-valued data to convert it into a "suitable" form compatible with homomorphic encryption. Any encoding must come with a matching decoding. Additionally, such an encoding must be homomorphic w.r.t addition and multiplication, and injective. Most importantly, any such encoding technique must be efficient and not hinder the efficiency of the underlying HE scheme.

The interest in HE-compatible encoding to process real/rational inputs efficiently is evident from a number of previous works e.g. [1,2,7,13,17]. In most of the RLWE (Ring Learning with Error) hardness-based homomorphic encryption schemes a plaintext is viewed as an element of the ring $R_t = \mathbb{Z}_t[x]/\Phi_m(x)$ where $\Phi_m(x)$ is the m-th cyclotomic polynomial and $\mathbb{Z}_t$ is the ring of integers modulo t. Encoding integer input to a polynomial in R_t is relatively straightforward, namely one can consider the base t representation of the integer. For allowing integer and rational inputs one must define an encoding converting elements of $\mathbb{Z}$ or $\mathbb{Q}$ (typically represented as fixed-point decimal numbers in applications) into elements of R_t. Previous works [3,4,6,10,11,21,27] have proposed several encoding methods for integers and rationals. One previously taken approach is to scale the fixed-point numbers to integers and then encode them as polynomials (using a suitable base). Another approach is to consider them as fractional numbers. In [10] it was shown that these two representations are isomorphic. As pointed out in [10] the latter approach, although avoids the overhead of bookkeeping homomorphic ciphertext, is difficult to analyse.

All of the aforementioned encodings share a problem (discussed in [4, 10]); namely, t must have sufficiently large value for the encoding to work correctly. This large value of t means one may need to choose large parameters for the overall homomorphic encryption scheme hindering the efficiency. A clever solution to this problem was proposed by Chen, Len, Player and Xia [4], which borrows a mathematical technique from Hoffstein and Silverman [16] and combines it with the homomorphic encryption scheme proposed by Fan and Vercauteren [14]. The main idea of the so-called CLPX encoding [4] is to replace the modulus t with the polynomial $x - b$ for some positive integer b and turning the plaintext space into the quotient ring $\mathbb{Z}/(b^n + 1)\mathbb{Z}$. Note that CLPX encoding converts fractional or fixed-point numbers and the scheme combines it with a modified version (which we will call ModFV) of the original FV scheme.

In the CLPX encoding, the rational (input) domain is a finite subset of $\mathbb{Q}$ and, therefore, is not closed under the usual compositions (addition and multiplication) which can potentially lead to overflow problems. That is, if the composition of two rational inputs lies outside the domain then its decoding (after homomorphic computation) will be incorrect. However, they do not provide any analytical discussion or solution towards solving this problem. The theory behind our encoding, which also transforms fixed-point (decimal) numbers, allows us to provide an analytical solution to this problem.

1.1 Our Results

The main aim of our work is to investigate an efficient HE-compatible encoding method that is generic (not necessarily targeted for a specific HE scheme) and can be easily adapted to apply to the HE schemes over integers or polynomials. The results of this work are as follows:

- We construct an efficient and generic encoding (and decoding) scheme based on a transformation that stems from p-adic number theory. First, we identify the tools and techniques provided by p-adic number theory to derive the foundational injective transformation that maps rationals to (modulo) integers, and is additively and multiplicatively homomorphic. The encoding scheme follows naturally from this injective transformation. We call this new encoding PIE ($\underline{p}$-ad$\underline{i}$c $\underline{e}$ncoding).
- We use the structural properties of the rational domain (of the above-mentioned transformation) to provide a bound on the domain size ensuring that there is no overflow from (additive or multiplicative) composition thus causing incorrect decoding. The previous work [4], did not address this overflow problem.
- Finally, we demonstrate that our encoding map can be easily applied to both Approximate Greatest Common Divisor(AGCD)-based and RLWE-based (over polynomial rings) HEs using the Batch FHE [5] and the modified Fan-Vercauteren(ModFV) [4] schemes respectively. We also discuss how the (public) parameters of these HEs can be used to setup the parameters of PIE.

We show our encoding scheme allows for a much larger input space compared to the previous encoding scheme [4] for an RLWE-based HE without severely compromising the circuit depth that can be evaluated using the HE. To the best of our knowledge this is the first work discussing an encoding scheme for AGCD-based schemes.

We implemented PIE using C++ (together with proof-of-concept implementations of IDGHV (Batch FHE) [5] and ModFV schemes[1]) to estimate the efficiencies of the encoding and decoding. Our implementation can be found at `https://github.com/Algemetric/pie-cpp`. The results of our experiment are given in Section 6.

2 Notations and Foundations

In this section we introduce the basic ideas and techniques from p-adic number theory that are necessary for developing our encoding scheme. We emphasize that the ideas described in this section are self-contained and do not assume prior knowledge of p-adic number theory.

[1] FHE part of our implementation is not optimized

2.1 Notations

For a real number r, the functions $\lfloor r \rfloor$, $\lceil r \rceil$, $\lfloor r \rceil$ denote the usual "floor", "ceiling", and "round-to-nearest-integer" functions. For an integer a, $|a|_{\text{bits}}$ denotes the bit length of a. The ring of integers is denoted by $\mathbb{Z}$, and the field of rationals by $\mathbb{Q}$. For a positive integer n, $\mathbb{Z}/n\mathbb{Z}$ denotes the ring of integers modulo n. In case n is prime, we sometimes write $\mathbb{F}_n$. To distinguish this ring (field) from sets of integer representatives, we denote by $\mathcal{Z}_n$ the set $\big[-\lceil (n-1)/2 \rceil, \lfloor (n-1)/2 \rfloor\big] \cap \mathbb{Z}$. For integers a, n we denote by $a \bmod n$ the unique integer $\overline{a} \in \mathcal{Z}_n$ such that $n \mid (a - \overline{a})$. Similarly, we use the elements of $\mathcal{Z}_n$ as representatives of the cosets of $\mathbb{Z}/n\mathbb{Z}$, and sometimes use $\mathcal{Z}_n$ in place of $\mathbb{Z}/n\mathbb{Z}$, though in this case we are careful to put "$\bmod n$" where appropriate. For a polynomial p, $\lfloor \mathsf{p} \rceil$ and $[\mathsf{p}]_n$ denote the rounding of each coefficient to the nearest integer, and the reduction of each coefficient modulo n. We use everywhere $\log(\cdot)$ in place of $\log_2(\cdot)$. "Input space" will always mean the set of fractions for which encoding correctness holds, and "message space" always means a subset of the input space for which homomorphic correctness (for arithmetic circuits up to a certain depth) holds.

2.2 Results and Techniques from *p*-adic Arithmetic

Roughly speaking, p-adic number theory allows us to represent a rational $\frac{x}{y} \in \mathbb{Q}$ using integers. If $\frac{x}{y} \in \mathbb{Q}$ and p is a prime then we have

$$\frac{x}{y} = \sum_{j=n}^{\infty} a_j p^j = a_n p^n + a_{n+1} p^{n+1} + \dots, \tag{1}$$

where $0 \leq a_j < p$ and $n \in \mathbb{Z}$. When $n \in \mathbb{Z}^+ \cup \{0\}$ the sum in 1 is called a p-adic integer. Equivalently, observe that any rational x/y can be rewritten in the form

$$\frac{x}{y} = \frac{x'}{y'} p^v, \text{ where } \gcd(x', p) = \gcd(y', p) = 1$$

The number v is called the p-adic valuation of x/y. In case $v \geq 0$, x/y is a p-adic integer. The ring of p-adic integers is denoted by $\mathbb{Z}_p$.

An r-segment p-adic representation, a.k.a. Hensel code, simply truncates the above sum after $j = r - 1$. In this case, the power series in eq. (1) becomes

$$\sum_{j=n}^{r-1} a_j p^j + O(p^r).$$

A natural consequence of this truncated representation is a mapping (discussed in detail in Definition 3) from a set of rationals to $\mathbb{Z}/p^r\mathbb{Z}$. This mapping is the main component of our encoding scheme.

A specific set of rational numbers (p-adic numbers) called the Farey rationals are defined as follows.

Definition 1 (Farey rationals [15]). *Given a prime p and an integer $r \geq 1$, let $N = \left\lfloor \sqrt{\frac{p^r-1}{2}} \right\rfloor$. The Farey rationals are defined as*

$$\mathcal{F}_N = \left\{ \frac{x}{y} : 0 \leq |x| \leq N, 1 \leq y \leq N \right\} \tag{2}$$

where $\gcd(x, y) = \gcd(y, p) = 1$.

We note that every rational in $\mathcal{F}_N$ has p-adic valuation $v \geq 0$, and therefore $\mathcal{F}_N \subset \mathbb{Z}_p$; i.e. every Farey rational is a p-adic integer.

For describing the mapping on which our encoder is based, we need to introduce the modified extended Euclidean algorithm MEEA [18–20, 22, 23, 26]. The MEEA is simply a truncated version of the extended Euclidean algorithm (EEA) and is similarly efficient. We pause briefly to describe the EEA. Recall that the EEA calculates the greatest common divisor of two integers x_0, x_1 along with the associated Bézout coefficients $y, z \in \mathbb{Z}$ such that $x_0 \cdot y + x_1 \cdot z = \gcd(x_0, x_1)$. The computation generates the tuples $(x_2, \ldots, x_n)$, $(y_2, \ldots, y_n)$, $(z_2, \ldots, z_n)$, and $q_i = \lfloor x_{i-1}/x_i \rfloor$ such that:

$$\begin{aligned} x_{i+1} &= x_{i-1} - q_i x_i, \quad && \text{where } x_0,\ x_1 \text{ are the input,} \\ y_{i+1} &= y_{i-1} - q_i y_i, && \text{with } y_0 = 0,\ y_1 = 1, \\ z_{i+1} &= z_{i-1} - q_i z_i, && \text{with } z_0 = 1,\ z_1 = 0. \end{aligned}$$

Moreover, for each $i \leq n$, we have $y_i x_1 + z_i x_0 = x_i$. The computation stops with $x_n = 0$, at which point $x_{n-1} = \gcd(x_0, x_1)$.

Definition 2 (MEEA, [18]). *Given $x_0, x_1 \in \mathbb{Z}$, $\mathsf{MEEA}(x_0, x_1)$ is defined as the output $(x, y) = \left((-1)^{i+1} x_i,\ (-1)^{i+1} y_i\right)$ of the extended Euclidean algorithm (as described above) once $|x_i| \leq N$.*

Now we are ready to define the necessary mapping from $\mathcal{F}_N$ to $\mathcal{Z}_{p^r}$.

Definition 3 ([28]). *The mapping $H_{p^r} : \mathcal{F}_N \to \mathcal{Z}_{p^r}$ and its inverse are defined as*

$$H_{p^r}\left(\frac{x}{y}\right) = xy^{-1} \bmod p^r, \tag{3}$$

$$H_{p^r}^{-1}(h) = \mathsf{MEEA}(p^r, h) \tag{4}$$

The H-mapping is injective and, therefore, gives a unique representation [28] of each element of $\mathcal{F}_N$ in $\mathcal{Z}_{p^r}$. The inverse of H_{p^r} is well-defined.

Proposition 1. *For all $x/y \in \mathcal{F}_N$ and $h \in H_{p^r}(\mathcal{F}_N) \subseteq \mathcal{Z}_{p^r}$,*

(i) $H_{p^r}^{-1}\left(H_{p^r}(x/y)\right) = x/y$, *and*
(ii) $H_{p^r}\left(H_{p^r}^{-1}(h)\right) = h$.
(iii) *If $a, a' \in \mathbb{Z}$ and $a' = a \pmod{p^r}$, then $H_{p^r}^{-1}(a) = H_{p^r}^{-1}(a')$.*

Proof. (i) Let $x/y \in \mathcal{F}_N$, $H_{p^r}\left(x/y\right) = h$, and suppose $H_{p^r}^{-1}(h) = a/b$. By definition of the MEEA and $H_{p^r}^{-1}$, there is an integer c such that $bh + cp^r = a$. But then $b(xy^{-1}) \equiv a \pmod{p^r}$, which implies $xy^{-1} \equiv ab^{-1} \pmod{p^r}$. That is, $H_{p^r}\left(a/b\right) = H_{p^r}\left(x/y\right)$. That $a/b = x/y$ then follows from injectivity of H_{p^r}.

(ii) Let $h \in H_{p^r}\left(\mathcal{F}_N\right) \subsetneq \mathcal{Z}_{p^r}$, and suppose $H_{p^r}^{-1}(h) = x/y$. By definition of the MEEA, there is an integer z such that $yh + zp^r = x$. Clearly $xy^{-1} \equiv h \pmod{p^r}$, proving the result.

(iii) Let $h' = h + kp^r$, the $\mathsf{MEEA}(p^r, h')$ generates tuples $(x'_0, x'_1, x'_2, x'_3, \ldots) = (p^r, h', p^r, h, \ldots)$ and $(y'_0, y'_1, y'_2, y'_3, \ldots) = (0, 1, 0, 1, \ldots)$. Whereas running MEEA with p^r and h generates tuples $(x_0, x_1, \ldots) = (p^r, h, \ldots)$ and $(y_0, y_1, \ldots) = (0, 1, \ldots)$. Notice that $x'_2 = x_0$ and $y'_2 = y_0$. An easy induction shows that $x'_i = x_{i-2}$ and $y'_i = y_{i-2}$, for $i = 2, 3, \ldots$, whence $\mathsf{MEEA}(p^r, h') = \mathsf{MEEA}(p^r, h)$. This completes the proof.

Proposition 2. *The mappings H_{p^r} and $H_{p^r}^{-1}$ are homomorphic w.r.t. addition and multiplication in the following sense.*

(i) *For all $u, u' \in \mathcal{F}_N$, $H_{p^r}(u) \cdot H_{p^r}(u') = H_{p^r}(u \cdot u') \bmod p^r$ and $H_{p^r}(u) + H_{p^r}(u') = H_{p^r}(u + u') \bmod p^r$.*

(ii) *If $h, h' \in \mathbb{Z}_{p^r}$ and $H_{p^r}^{-1}(h) \cdot H_{p^r}^{-1}(h'), H_{p^r}^{-1}(h) + H_{p^r}^{-1}(h') \in \mathcal{F}_N$, then $H_{p^r}^{-1}(h \cdot h') = H_{p^r}^{-1}(h) \cdot H_{p^r}^{-1}(h')$ and $H_{p^r}^{-1}(h + h') = H_{p^r}^{-1}(h) + H_{p^r}^{-1}(h')$.*

Proof. (i) Let $a/b, c/d \in \mathcal{F}_N$. By definition of the Farey rationals, a, b, c, d are co-prime with p. That H_{p^r} is homomorphic with respect to addition and multiplication follows from the properties of congruences:

$$ac(bd)^{-1} = (ab^{-1})(cd^{-1}) \bmod p^r$$
$$\text{and } (ad + bc)(bd)^{-1} = (ab^{-1} + cd^{-1}) \bmod p^r.$$

(ii) Invoking the homomorphic property of H_{p^r}, from $H_{p^r}^{-1}(h) \cdot H_{p^r}^{-1}(h'), H_{p^r}^{-1}(h) + H_{p^r}^{-1}(h') \in \mathcal{F}_N$ we obtain $h \cdot h', h + h' \in H_{p^r}(\mathcal{F}_N)$. By proposition 1(ii), $H_{p^r}\left(H_{p^r}^{-1}(h \cdot h')\right) = h \cdot h'$ and $H_{p^r}\left(H_{p^r}^{-1}(h + h')\right) = h + h'$. The result follows from the injectivity of H_{p^r}.

Example 1. Given rationals $a = 12.37$ and $b = 8.3$, we choose the $p = 3, r = 10$. Here $N = \left\lfloor \sqrt{(p^r - 1)/2} \right\rfloor = 125261$. We compute the encodings of a and b as h_1 and h_2:

$$h_1 = H_{p^r}\left(\tfrac{1237}{100}\right) = 2196674185$$
$$h_2 = H_{p^r}\left(\tfrac{83}{10}\right) = 9414317891$$

We can now compose the rationals with addition, subtraction, and multiplication, and decode to check correctness:

$$\begin{aligned} r_1 &= h_1 + h_2 \bmod p^r = 11610992076, & H_{p^r}^{-1}(r_1) &= \tfrac{2067}{100} &&= 20.67 \\ r_2 &= h_1 - h_2 \bmod p^r = 24163415903, & H_{p^r}^{-1}(r_2) &= \tfrac{407}{100} &&= 4.07 \\ r_3 &= h_1 h_2 \bmod p^r \quad = 2541865931, & H_{p^r}^{-1}(r_3) &= \tfrac{102671}{1000} &&= 102.671 \end{aligned}$$

Choice of p and r At this point it is clear that the H-mapping (in definition 3) can be used to map a set of Farey rationals into $\mathcal{Z}_{p^r}$. Thus, it can be used for encoding rational data that are contained in the set of Farey rationals which is the domain of the mapping. A natural question is: given a set of rationals how to choose p^r (and, therefore, N) so that $\mathcal{F}_N$ contains the rationals one wishes to encode? We point out that for a finite set of rationals $\mathcal{S}$, one can choose $p^r \geq \max_{a,b:a/b\in\mathcal{S}}(2a^2+1, 2b^2+1)$. Choosing a small p and a very large r is possible, though this could restrict the number of rationals that can be mapped due to the gcd condition (in definition 1). We illustrate this with examples in Appendix A.

Replacing the prime power with a composite. The above results can be extended when p^r is replaced by an arbitrary positive integer g. Let $p_1, \ldots, p_k$ be distinct primes, $g = p_1^{r_1} \cdots p_k^{r_k}$, and $N = \left\lfloor \sqrt{(g-1)/2} \right\rfloor$. The Farey rationals defined by g are simply the set of reduced fractions

$$\mathcal{F}_N = \left\{ \frac{x}{y} \,\middle|\, 0 \leq |x| \leq N,\ 1 \leq y \leq N, \gcd(y,g) = 1,\ \gcd(x,y) = 1 \right\}.$$

We briefly recall (the integer version of) the Chinese Remainder Theorem (CRT), as it is necessary for our encoding scheme.

Definition 4 (Chinese Remainder Theorem). *Let $n_1, \ldots, n_k$ be k co-prime integers, and $n = \prod_{i=1}^{k} n_i$. The CRT describes the isomorphism $\mathbb{Z}/n\mathbb{Z} \cong \mathbb{Z}/n_1\mathbb{Z} \times \cdots \times \mathbb{Z}/n_k\mathbb{Z}$ given by*

$$x \mapsto (x \bmod n_1, \ldots, x \bmod n_k).$$

We denote the x such that $x = h_i \bmod n_i$ and $(h_1, \ldots, h_k) \in \mathbb{Z}/n_1\mathbb{Z} \times \cdots \times \mathbb{Z}/n_k\mathbb{Z}$ by $\mathsf{CRT}_{n_1,\ldots,n_k}(h_1, \ldots, h_k)$.

Remark 1. In the following definition, we abuse notation slightly and identify $\mathsf{CRT}_{\ldots}(\ldots)$ not with actual ring elements in $\mathbb{Z}/n\mathbb{Z}$, but with integer representatives in $\mathcal{Z}_n$.

Definition 5 ([24,25]). *The injective mapping $H_g : \mathcal{F}_N \to \mathcal{Z}_g$ and its inverse are defined as*

$$H_g(x/y) = \mathsf{CRT}_{p_1^{r_1},\ldots,p_k^{r_k}}\left(H_{p_1^{r_1}}(x/y), \ldots, H_{p_k^{r_k}}(x/y)\right) \tag{5}$$

$$H_g^{-1}(h) = \mathsf{MEEA}(g, h) \tag{6}$$

The following proposition is an extension of proposition 1 for composite g and its proof proceeds similar to the proof of proposition 1.

Proposition 3. *Let $N = \left\lfloor \sqrt{(g-1)/2} \right\rfloor$. For all $x/y \in \mathcal{F}_N$ and $h \in H_g(\mathcal{F}_N) \subsetneq \mathcal{Z}_g$,*

(i) $H_g^{-1}(H_g(x/y)) = x/y$, *and*
(ii) $H_g\left(H_g^{-1}(h)\right) = h$.
(iii) *If $h, h' \in \mathbb{Z}$ and $h' = h \pmod g$, then $H_g^{-1}(h) = H_g^{-1}(h')$.*

Proposition 4. *The mapping H_g is homomorphic w.r.t. addition and multiplication, and H_g^{-1} is homomorphic as in proposition 2.*

Proof. Let $N = \left\lfloor \sqrt{(g-1)/2} \right\rfloor$, and $u, u' \in \mathcal{F}_N$. Using the homomorphic properties of the CRT where necessary, we have

$$H_g(u+u') = \mathsf{CRT}_{p_1^{r_1},\ldots,p_k^{r_k}}\left(H_{p_1^{r_1}}(u+u'), \ldots, H_{p_k^{r_k}}(u+u')\right),$$

and

$$H_g(u) + H_g(u') \\ = \mathsf{CRT}_{p_1^{r_1},\ldots,p_k^{r_k}}\left(H_{p_1^{r_1}}(u) + H_{p_1^{r_1}}(u'), \ldots, H_{p_k^{r_k}}(u) + H_{p_k^{r_k}}(u')\right).$$

By proposition 2(i), each $H_{p_i^{r_i}}(u) + H_{p_i^{r_i}}(u') = H_{p_i^{r_i}}(u+u') \bmod p_i^{r_i}$. Whence $H_g(u+u') = H_g(u) + H_g(u')$. The proof that $H_g(u \cdot u') = H_g(u) \cdot H_g(u')$ is analogous.

To establish the homomorphic properties of H_g^{-1} simply replace p^r by g everywhere in the proof of proposition 2(ii). $\square$

Example 2. Suppose we have the same rationals of Example 1: $a = 12.37$ and $b = 8.3$. We now choose the $p = 6, r = 17$ and $g = p^r + 1 = 16926659444737$, which yields $N = \left\lfloor \sqrt{(g-1)/2} \right\rfloor = 2909180$. The encodings of a and b are

$$h_1 = H_g\left(\tfrac{1237}{100}\right) = 16757392850302$$
$$h_2 = H_g\left(\tfrac{83}{10}\right) = 1692665944482.$$

Again, we compose the encodings, and verify the correctness of the results:

$$r_1 = h_1 + h_2 \bmod g = 1523399350047, \quad H_g^{-1}(r_1) = \tfrac{2067}{100} = 20.67$$
$$r_2 = h_1 - h_2 \bmod g = 15064726905820, \quad H_g^{-1}(r_2) = \tfrac{407}{100} = 4.07$$
$$r_3 = h_1 h_2 \bmod g = 7058416988558, \quad H_g^{-1}(r_3) = \tfrac{102671}{1000} = 102.671$$

Remark 2. Definitions 3 and 5 coincide when $g = p^r$ (a prime power), so one should take the latter as the general definition of H and H^{-1}, picking g to be a prime power when necessary.

Size of the set. The cardinality of $\mathcal{F}_N$ for $N = \lfloor\sqrt{(g-1)/2}\rfloor$ depends heavily on the choice of g. This is because the number of fractions x/y with $|x|, |y| \leq N$ that fail the condition $\gcd(y, g) = 1$ depends on the prime factorization of g – the more "small" prime factors g has, the more fractions fail the gcd condition.

Proposition 5. *The cardinality of $\mathcal{F}_N$ for $N = \lfloor\sqrt{(g-1)/2}\rfloor$ is given by*

$$4 \cdot \Phi(N) + 1 - \big(\# \text{ of } x/y \text{ with } \gcd(y, g) \neq 1\big)$$

where $\Phi(k) = \sum_{i=1}^{k} \phi(i)$, and ϕ is the Euler's totient function.

Proof. Use the fact that the k^{th} Farey *sequence*[2] has length $1 + \Phi(k)$, and then enforce the gcd condition on the Farey rationals.

Simulations show that when g is an odd prime,

$$|\mathcal{F}_N| = 4 \cdot \Phi(N) + 1 \approx 0.6g. \tag{7}$$

This fact will be used for comparison with existing work in section 5.2.

3 PIE: A Rational Encoder

Let g be a positive integer, $N = \left\lfloor\sqrt{(g-1)/2}\right\rfloor$, and make $\mathcal{F}_N$ the input space. We define encoding and decoding as follows:

$\mathsf{PIE.Encode}(x/y)$. For $x/y \in \mathcal{F}_N$ output $H_g(x/y)$.

$\mathsf{PIE.Decode}(z)$. For $z \in \mathcal{Z}_g$, output $H_g^{-1}(z)$.

Proposition 6. *For all $m, m' \in \mathcal{F}_N$ such that $m \cdot m' \in \mathcal{F}_N$,*

$$\mathsf{PIE.Decode}\left([\mathsf{PIE.Encode}(m) \cdot \mathsf{PIE.Encode}(m') \bmod g]\right) = m \cdot m'$$

and $\forall m, m' \in \mathcal{F}_N$ s.t. $m + m' \in \mathcal{F}_N$

$$\mathsf{PIE.Decode}\left([\mathsf{PIE.Encode}(m) + \mathsf{PIE.Encode}(m') \bmod g]\right) = m + m'$$

Proof. Use proposition 3(i), proposition 3(iii), and proposition 4.

Corollary 1. *Let p be a multivariate polynomial with coefficients in $\mathbb{Q}$. For all $m_0, \ldots, m_k \in \mathcal{F}_N$ such that $\mathsf{p}(m_0, \ldots, m_k) \in \mathcal{F}_N$,*

$$\mathsf{PIE.Decode}\Big(g, \mathsf{p}\big(\mathsf{PIE.Encode}(g, m_0), \ldots, \mathsf{PIE.Encode}(g, m_k)\big) \bmod g\Big)$$
$$= \mathsf{p}(m_0, \ldots, m_k).$$

[2] The k^{th} Farey sequence is the set of reduced fractions in the interval $[0, 1)$ with numerator and denominator each at most k.

As indicated in the preceding results, for the encoding (and decoding) to yield the correct result when used in an HE scheme, one must ensure that if two or more elements from $\mathcal{F}_N$ are combined using additions and/or multiplications then any intermediates and the final output must not lie outside the set $\mathcal{F}_N$. For this reason we will define the (rational) message space to be the following subset of $\mathcal{F}_N$:

$$\mathcal{G}_M = \{x/y \in \mathcal{F}_N : 0 \leq |x| \leq M, 1 \leq y \leq M\} \tag{8}$$

The main idea behind choosing a subset of $\mathcal{F}_N$ as the set of messages is that when elements from $\mathcal{G}_M$ are combined, the resulting element can be in $\mathcal{F}_N$. Ensuring the output lands in $\mathcal{F}_N$ induces a bound on the number of computations that can be performed, and determines the choice of parameters involved therein. At this point, one might wonder whether we need to do something similar with the range $\mathcal{Z}_g$ of the encoder to make sure that overflow modulo g does not occur during computations. The answer is "no". This is because proposition 3(iii) along with the above message space restriction imply that overflow modulo g does not affect decoding.

The choice of M depends jointly on the rational data one must encode, and the circuits one must evaluate over those data. We elaborate this in the following section.

3.1 Choosing the Message Space $\mathcal{G}_M$.

We will follow closely the analysis of van Dijk et al. in section 3.2 of [12], and describe an arithmetic circuit in terms of the multivariate polynomial it computes. To this end, recall that the ℓ_1-norm of a polynomial is simply the sum of the absolute values of its coefficients.

Polynomials with which PIE *is compatible.* Let $\mathcal{P}_{d,t}$ denote the set of polynomials in $\mathbb{Q}[x_1, x_2, \ldots]$ with total degree at most d and ℓ_1-norm at most t, whose coefficients have absolute value at least 1. For example, $\mathcal{P}_{d,t}$ contains polynomials of the form

$$\mathsf{p}(x_1, \ldots, x_k) = \sum_{d_1+\cdots+d_k \leq d} \sum_{\alpha=1}^{I} c_\alpha x_1^{d_1} x_1^{d_2} \cdots x_k^{d_k},$$

where each $|c_\alpha| \geq 1$, and $\sum_\alpha |c_\alpha| \leq t$.
The following proposition establishes an upper bound on the output of a polynomial in $\mathcal{P}_{d,t}$ when all inputs are from $\mathcal{G}_M$.

Proposition 7. *If $x_1/y_1, \ldots, x_k/y_k \in \mathcal{G}_M$, $\mathsf{p} \in \mathcal{P}_{d,t}$ is k-variate, and $\mathsf{p}(x_1/y_1, \ldots, x_k/y_k) = x/y$, then*

$$|x| \leq t \cdot M^{dt} \text{ and } |y| \leq M^{dt}$$

Proof. Note that $\mathsf{p} \in \mathcal{P}_{d,t}$ can be written as $\mathsf{p} = \sum_i c_i \mathsf{p}_i$, where $\sum_i |c_i| \leq t$, each $|c_i| \geq 1$, and each p_i is a monomial of degree at most d.
Let $\mathsf{p} = \sum_{i=1}^{I} c_i \mathsf{p}_i$.
Since $\deg(\mathsf{p}_i) \leq d$, the evaluation $\mathsf{p}_i\big(x_1/y_1, \ldots, x_k/y_k\big)$ is a fraction of the form

$$\frac{a_i}{b_i} = \frac{x_{i_1} x_{i_2} \cdots x_{i_\ell}}{y_{i_1} y_{i_2} \cdots y_{i_\ell}}, \text{ for some } \ell \leq d \text{ and } \{i_1, \ldots, i_\ell\} \subseteq \{1, \ldots, k\}.$$

As each $x_i/y_i \in \mathcal{G}_M$, we have $|a_i|, |b_i| \leq M^\ell \leq M^d$.
Since $x/y = \sum_{i=1}^{I} c_i \cdot a_i/b_i$, there are nonzero integers α and β such that

$$\alpha x = (c_1 a_1) b_2 b_3 \cdots b_I + b_1 (c_2 a_2) b_3 \cdots b_I + b_1 b_2 \cdots b_{I-1} (c_I a_I)$$
$$\text{and } \beta y = b_1 b_2 \cdots b_I.$$

It follows from $\sum |c_i| \leq t$ and the above bound on $|a_i|, |b_i|$ that

$$|x| \leq \sum_{i=1}^{I} |c_i| (M^d)^I \leq t \cdot M^{dI} \text{ and } |y| \leq M^{dI}.$$

The proof is completed by observing that $|c_i| \geq 1$, for all i, implies $I \leq t$.

Proposition 8. *A sufficient condition for compatibility of* PIE *with polynomials in* $\mathcal{P}_{d,t}$ *as in Corollary 1:*

$$\mathsf{PIE.Decode}\Big(g, \mathsf{p}\big(\mathsf{PIE.Encode}(g, m_0), \ldots, \mathsf{PIE.Encode}(g, m_k)\big) \bmod g\Big)$$
$$= \mathsf{p}(m_0, \ldots, m_k)$$

is

$$M \leq \left(\frac{N}{t}\right)^{\frac{1}{dt}}, \text{ equivalently } d \leq \frac{\log(N) - \log(t)}{t \log(M)}. \tag{9}$$

Proof. Suppose M is chosen according to equation 9, and let $\mathsf{p} \in \mathcal{P}_{d,t}$ be k-variate. According to proposition 7, if $\mathbf{m} \in \mathcal{G}_M^k$ and $\mathsf{p}(\mathbf{m}) = x/y$, then

$$|x| \leq t \cdot M^{dt} \leq t \cdot \Big((N/t)^{\frac{1}{dt}} \Big)^{dt} = N, \text{ and} \tag{10}$$

$$|y| \leq M^{dt} \leq \Big((N/t)^{\frac{1}{dt}} \Big) = N/t \leq N. \tag{11}$$

Clearly $\gcd(g, y) = 1$, since y is a factor of the product of the denominators in $\mathbf{m}$. Thus $\mathsf{p}(\mathbf{m}) \in \mathcal{F}_N$, and the proof is completed.

4 PIE with a Batch FHE over Integers

Batch FHE [5] We briefly recall the scheme introduced by Cheon, Coron, Kim, Lee, Lepoint, Tibuchi and Yun [5], following their notations. Let λ be the security parameter, γ and η be the bit-length of the public and secret key respectively, and ρ be the bit-length of noise. Further, choose ℓ_Q-bit integers $Q_1, \ldots, Q_\ell$. The IDGHV scheme is defined as follows.

$\mathsf{IDGHV.KGen}(1^\lambda, (Q_j)_{1\le j\le \ell})$. Choose distinct η-bit primes $p_1, \dots, p_\ell$, and let π be their product. Choose a uniform 2^{λ^2}-rough[3] integer $q_0 < 2^\gamma/\pi$, and let the public key be $x_0 = q_0 \cdot \pi$. It is required that $\gcd(\prod_j Q_j, x_0) = 1$. Choose integers x_i, and x'_i with a quotient by π uniformly and independently distributed in $\mathbb{Z}\cap[0, q_0)$, and with the distribution of modulo p_j for $1 \le j \le \ell$ as follows

$$\begin{aligned} 1 \le i \le \tau,\ & x_i \bmod p_j = Q_j r_{i,j}, && r_{i,j} \leftarrow \mathbb{Z}\cap(-2^\rho, 2^\rho) \\ 1 \le i \le \ell,\ & x'_i \bmod p_j = Q_j r'_{i,j} + \delta_{i,j}, && r'_{i,j} \leftarrow \mathbb{Z}\cap(-2^\rho, 2^\rho) \end{aligned}$$

Let $\mathsf{pk} = \{x_0, (Q_i)_{1\le i\le\ell}, (x_i)_{1\le i\le\tau}, (x'_i)_{1\le i\le\ell}\}$ and
$\mathsf{sk} = (p_j)_{1\le j\le\ell}$

$\mathsf{IDGHV.Enc}(\mathsf{pk}, \mathbf{m})$. For $\mathbf{m} = (m_1, \dots, m_\ell) \in \mathbb{Z}/Q_1\mathbb{Z} \times \dots \times \mathbb{Z}/Q_\ell\mathbb{Z}$, choose a random binary vector $\mathbf{b} = (b_1, \dots, b_\tau)$ and output the ciphertext

$$c = \left(\sum_{i=1}^{\ell} m_i \cdot x'_i + \sum_{i=1}^{\tau} b_i \cdot x_i\right) \pmod{x_0}$$

$\mathsf{IDGHV.Dec}(\mathsf{sk}, c)$. $\mathbf{m} = (m_1, \dots, m_\ell)$ where $m_j \leftarrow c \bmod p_j \pmod{Q_j}$

$\mathsf{IDGHV.Add}(\mathsf{pk}, c_1, c_2)$. Output $c_1 + c_2 \bmod x_0$

$\mathsf{IDGHV.Mult}(\mathsf{pk}, c_1, c_2)$. Output $c_1 \cdot c_2 \bmod x_0$

The security of the IDGHV scheme is based on the decisional approximate GCD problem (DACD) [5].

4.1 PIE with IDGHV

Permitted circuits and Parameters for IDGHV.

Definition 6 ([8]). *Let C be an arithmetic circuit and $\rho' = \max\{\rho + \log(\ell) + \ell_Q, 2\rho + \log(\tau)\}$. C is a* permitted circuit *if every input being bounded in absolute value by $2^{\rho'+\ell_Q}$ implies the output is bounded in absolute value by $2^{\eta-4}$.*

Describing circuits in terms of the multivariate polynomial they compute yields a sufficient condition for determining whether a given circuit is permitted.

Lemma 1 ([8]). *Let C be an arithmetic circuit over the rationals comprised of addition/subtraction and multiplication gates, f be the multivariate polynomial that C computes, and $|f|_1$ be the ℓ_1 norm of f. If*

$$\deg(f) < \frac{\eta - 4 - \log\left(|f|_1\right)}{\rho' + \ell_Q},$$

then C is a permitted circuit.

[3] An integer is *b-rough* provided it has no prime factors smaller than b.

One can show that for a circuit with multiplicative depth D, the total degree of the polynomial f computed by the circuit is at most $2^{D-1}+1 \approx 2^{D-1}$. Further, we note that maximum value of $\deg(f)$ is (roughly) inversely proportional to $|Q_i|_{\text{bits}} = \ell_Q$, so the multiplicative depth of permitted circuits decreases as the bit size of the Q_i increases.

As in [8], we assume here that $\log(|f|_1) \ll \eta, \rho'$, so it suffices to choose ℓ, ℓ_Q such that $\eta/(\rho' + \ell_Q)$ is not too small. To this end, suppose we want to support circuits computing a polynomial of degree at most δ. Then we choose $\ell < 2^\rho$, $\ell_Q = O(\rho)$, and $\eta \geq \rho' \Theta(\delta)$. In particular, we recommend:

$$\ell \ll 2^\rho,\ \ell_Q \approx \rho,\ \text{and}\, \eta = 3\rho'\delta.$$

Parameters for PIE with IDGHV. The maximum depth of circuits with which PIE is compatible depends on the size of the message space $\mathcal{G}_M$ relative to the size of the input space $\mathcal{F}_N$ (i.e. how small M is relative to N). This means that fixing M determines the circuits one can evaluate, and fixing the circuits to be evaluated determines M. We give an analytical discussion of the two cases below.
First, we pause to remind the reader of the relevant parameter sizes for IDGHV. For ciphertexts of the form $c = \mathsf{CRT}_{q_0,p_1,\ldots,p_\ell}(q, Q_1 r_1 + m_1, \ldots, Q_\ell r_\ell + m_\ell)$, we have $|p_i|_{\text{bits}} = \eta$, $|Q_i|_{\text{bits}} = \ell_Q$, and $\rho' = \max\{\rho + \log(\ell) + \ell_Q, 2\rho + \log(\tau)\}$.

In the following discussion, $g = \prod Q_i$, $N = \left\lfloor \sqrt{(g-1)/2} \right\rfloor$, and $\mathcal{G}_M$ is the message space, where $M \leq N$.

Choosing circuits first. Given a set of circuits, we must choose d and t so that $\mathcal{P}_{d,t}$ contains the polynomials which the circuits in the set compute. To this end, choose d, t to satisfy lemma 1. That is,

$$d < \frac{\eta - 4 - \log(t)}{\rho' + \ell_Q}.$$

We put $t = 1$ for convenience and to maximize the multiplicative depth of permitted circuits, whence the permitted circuits are given by $\mathcal{P}_{d,1}$ for $d \approx (\eta - 4)/(\rho' + \ell_Q) - 1$. Rewriting eq. (9) to get a bound on $|M|_{\text{bits}}$ and using the above values of d, t we obtain

$$|M|_{\text{bits}} \approx \frac{\ell \ell_Q \rho' + \ell_Q^2}{2(\eta - \rho' - \ell_Q - 4)} \tag{12}$$

Note that t may be chosen much larger, though too large a value may force M to be unreasonably small in order to satisfy eq. (9).

Choosing messages first. M must satisfy equation 9. Thus circuits which compute polynomials in $\mathcal{P}_{d,t}$ are permitted as long as

$$\frac{\log(t)}{\log(M)} + dt \leq \frac{\log(N)}{\log(M)}.$$

This inequality is satisfied by choosing

$$t < M \text{ and } d \leq \frac{\log(N) - \log(M)}{M\log(M)}.$$

Thus we may choose

$$t = M - 1 \text{ and } d \approx \frac{\ell\ell_Q - 2\log(M)}{2M\log(M)}$$

Note that this will require the values of ℓ and ℓ_Q to be quite large.
E.g. $M\log(M) \lesssim \ell\ell_Q$.

Two Encoding Options. There are two ways to combine PIE with IDGHV: using the Chinese Remainder Theorem, and component-wise. The former encodes single rationals, while the latter encodes vectors of rationals. Depending on the application an user can choose one of these two. We elaborate them below.

Encoding with the Chinese Remainder Theorem Choose the public parameters $Q_1, \ldots, Q_\ell$ to be distinct odd primes. Let $g = \prod_{i=1}^{\ell} Q_i$, $N = \left\lfloor \sqrt{(g-1)/2} \right\rfloor$, and $M \ll N$.

We use the Chinese Remainder Theorem (CRT) to convert the integer output of PIE.Encode to a vector of integers which is the input to IDGHV. We encode and decode with IDGHV as the underlying encryption scheme as follows:

IDGHV.Encode. For $m \in \mathcal{G}_M$, output
$\Big(\mathsf{PIE.Encode}(m) \bmod Q_1, \ldots, \mathsf{PIE.Encode}(m) \bmod Q_\ell\Big)$

IDGHV.Decode. For $(h_1, \ldots, h_\ell) \in \mathbb{Z}/Q_1\mathbb{Z} \times \cdots \times \mathbb{Z}/Q_\ell\mathbb{Z}$, compute
$h = \mathsf{CRT}_{Q_1,\ldots,Q_\ell}(h_1, \ldots, h_\ell)$, then output $\mathsf{PIE.Decode}(h)$.

Encoding and decoding above are computed with H_g and its inverse.

Choosing M for CRT *Encoding.* M must be chosen according to eq. (12). That is,

$$|M|_{\text{bits}} \approx \frac{\ell\ell_Q\rho' + \ell_Q^2}{2(\eta - \rho' - \ell_Q - 4)}$$

Encoding Component-wise Choose the public parameters $Q_1, \ldots, Q_\ell$ to be not-necessarily-distinct primes, and put $M_i \ll N_i = \left\lfloor \sqrt{(Q_i - 1)/2} \right\rfloor$. Using equation (9), we obtain $M_i \leq \big(N_i/t\big)^{1/dt}$, where d, t are chosen according to lemma 1. The encoding is as follows:

IDGHV.Encode. For $(m_1, \cdots, m_\ell) \in \mathcal{G}_{M_1} \times \cdots \times \mathcal{G}_{M_\ell}$,
output $\Big(\mathsf{PIE.Encode}(m_1), \ldots, \mathsf{PIE.Encode}(m_\ell)\Big)$

IDGHV.Decode. For $(h_1, \dots, h_\ell) \in \mathbb{Z}/Q_1\mathbb{Z} \times \cdots \times \mathbb{Z}/Q_\ell\mathbb{Z}$, output $\Big(\mathsf{PIE.Decode}(h_1), \dots, \mathsf{PIE.Decode}(h_\ell)\Big)$

In the component-wise encoding, for each i, $\mathsf{PIE.Encode}(h_i)$ and $\mathsf{PIE.Decode}(h_i)$ are computed with Q_i as the modulus, i.e., the encoding and decoding functions are H_{Q_i} and the corresponding inverses.

Choosing the M_i for Component-wise Encoding. Since we are encoding with primes Q_i instead of their product, it suffices here to make a minor change to eq. (12). Namely, we put $\ell = 1$. This yields

$$|M_i|_{\text{bits}} \approx \frac{\ell_Q \rho' + \ell_Q^2}{2(\eta - \rho' - \ell_Q - 4)}.$$

4.2 IDGHV-Compatible Encoding Parameters and Message Space

Parameters for (the CRT version of) PIE + IDGHV							
λ	ℓ	ℓ_Q	$\max d$	$\|M\|_{\text{bits}}$	γ	η	ρ
50	6	60	15	10	$\approx 5.3 \cdot 10^8$	4248	100
60	8	80	19	13	$\approx 1.3 \cdot 10^9$	6402	120
70	10	100	18	23	$\approx 3 \cdot 10^9$	9041	140

Table 1: Size of the elements of the rational message space $\mathcal{G}_M$ along with maximum degree d of compatible polynomials. Parameters chosen according to the recommendations in Section 3.2 of [5].

Remark 3. $|M|_{\text{bits}} = 23$ simply means that the message space is comprised by fractions whose numerators and denominators are up to 23 bits. Note that the co-primality restriction will not apply if M is smaller than every prime factor of $g = \prod Q_i$.

Choosing the Q_i appropriately We emphasize that PIE may be attached to IDGHV regardless of the choice of the Q_i. However, the input space $\mathcal{F}_N$ (of PIE) may be too small to be useful if the number *and* size of the Q_i are too small. In contrast, note that the Q_i can be small as long as there are "enough" of them. Similarly, if the number of Q_i is small, then their product should be quite large. As an example of the former, if $Q_i = 3$ for $i = 1, \dots, 5$, then the message space of IDGHV is (isomorphic to) $\mathbb{Z}/3^5\mathbb{Z}$. The encoding modulus for

PIE is $3^5 = 243$ which is co-prime with 10, so we can encode certain decimal numbers up to precision 2 such as $1.37 = \frac{137}{100}$.

We can use the parameters given in [5] to determine the size of each element in the corresponding message space by coupling PIE with IDGHV. Let $Q_1, \ldots, Q_\ell$ be distinct primes - public key elements in IDGHV. For encoding a single message, we take the product of all Q_i's as g and encode the rational message using g. In [5], four different configurations are provided: Toy, Small, Medium, and Large. In the Medium configuration, we have 138 56-bit Q_i's. This gives us a g of roughly 7728 bits with an N of roughly 3864 bits. In the Large configuration, we have 531 71-bit Q_i's. This gives us a g of length roughly 37701 bits with an N of roughly 18850 bits.

A large N resulting from (secure) HE parameters, is very advantageous. For example, if we take $N \approx 2^{18850}$ and $M = 2^{64} - 1$ (that allows fractions with numerators and denominators of up to 64 bits to be encoded), then we can use eq. (9) to find sets of polynomials $\mathcal{P}_{d,t}$ with which PIE is compatible. In this case, we get compatibility with polynomials in $\mathcal{P}_{2^4,2^4}$ (total degree and ℓ_1-norm at most 2^4) or with polynomials in $\mathcal{P}_{10,29}$ (total degree at most 10 and ℓ_1-norm at most 29). These sets of polynomials correspond to arithmetic circuits of (approximate) multiplicative depth 4 and 3, respectively. Of course if one chooses a smaller M, then the multiplicative depth of compatible circuits increases.

5 PIE with Modified Fan-Vercauteren HE

The modified FV scheme We give a brief description of a modification of the FV HE scheme [4] that is based on the decisional ring learning with errors (RLWE) problem. We refer the readers to [27], [14] for more details on RLWE. The main difference between the modified FV (ModFV) and FV is that the former encrypts integers while the latter encrypts polynomials. In particular, ModFV is obtained from FV by attaching the Hat Encoder as defined in [4]. We recall the encoder here.

Definition 7 (Hat Encoder, [4]). *Let $\|\cdot\|_\infty$ denote the polynomial infinity norm. For $m \in \mathbb{Z}/(b^n+1)\mathbb{Z}$, $b \geq 2$ and $n \geq 1$, let $\widehat{m}$ be the polynomial with lowest degree such that $\|\widehat{m}\|_\infty \leq (b+1)/2$ and $\widehat{m}(b) = m \bmod b^n + 1$. Such a polynomial always exists and has degree at most $n-1$.*

Roughly speaking, the Hat encoder takes the base-b expansion of m with coefficients in Z_{b^n+1}, and then replaces everywhere b by an unknown x to obtain the polynomial $\widehat{m}(x)$.

We are now ready to define ModFV. For n a power of 2 (typically at least 1024), denote the $2n^{\text{th}}$ cyclotomic ring of integers by $R = \mathbb{Z}[x]/(x^n+1)$, and let R_a denote the ring obtained by reducing the coefficients of R modulo a. The plaintext space is the ring $\mathcal{M} = \mathbb{Z}_{b^n+1}$, for $b \geq 2$, and the ciphertext space is product ring $R_q \times R_q$ for $q \gg b$. Let λ be the security parameter and χ be a discrete Gaussian distribution with standard deviation σ (typically $\sigma \approx 3.19$).

ModFV.SecretKeyGen. Sample $s \in R$ with coefficients uniform in $\{-1, 0, 1\}$.
Output $\mathsf{sk} = s$.

ModFV.PublicKeyGen(sk). Let $s = \mathsf{sk}$. Sample $a \leftarrow R_q$, and $e \leftarrow \chi$.
Output $\mathsf{pk} = \big([-(as + e)]_q, a\big) \in R_q \times R_q$.

ModFV.EvalKeyGen(sk). For $i = 0, \ldots, \ell$, where $w \geq 2$ and $\ell = \lfloor \log_w q \rfloor$, sample $a_i \leftarrow R_q$, and $e_i \leftarrow \chi$. Put $\mathsf{evk}[i] = \big([-(a_i s + e_i) + w^i s^2]_q, a_i\big) \in R_q \times R_q$.
Output the vector of pairs $\mathsf{evk} = \big(\mathsf{evk}[0], \ldots, \mathsf{evk}[\ell]\big)$.

ModFV.Enc(pk, $m \in \mathcal{M}$). Let $\Delta_b = \lfloor -\frac{q}{b^n+1}\left(x^{n-1} + bx^{n-2} + \ldots + b^{n-1}\right) \rceil$ and $\mathsf{pk} = (p_0, p_1)$. Sample $u \in R$ with coefficients uniform in $\{-1, 0, 1\}$, and $e_0, e_1 \leftarrow \chi$. Let $\widehat{m}$ be a hat encoding of m.
Output $\mathsf{ct} = \big([\Delta_b \widehat{m} + p_0 u + e_0]_q, [p_1 u + e_1]_q\big) \in R_q \times R_q$.

ModFV.Dec(sk, $\mathsf{ct} \in R_q \times R_q$). Let $s = \mathsf{sk}$ and $\mathsf{ct} = (c_0, c_1)$.
Let $\widehat{M} = \left\lfloor \frac{x-b}{q}[c_0 + c_1 s]_q \right\rceil$.
Output $m' = \widehat{M}(b) \in \mathcal{M}$.

5.1 PIE with ModFV

Since the CLPX encoding uses the same function H_g that defines our encoder, we follow closely the analysis presented by Chen et al. in Section 6.1 of [4]. We stress that although CLPX uses a function having the same definition as our "H-function", their approach is not based on techniques from p-adic number theory. Consequently, the decode functions and input spaces differ dramatically between CLPX and PIE. A comparison of the input spaces in provided in section 5.2

In pairing PIE with ModFV, we distinguish two cases: b^n+1 prime and b^n+1 composite. We note, however, that the definitions of encoding and decoding are identical for both cases. The differences lie in how b and n are chosen, and the resulting input spaces.

Put $N = \left\lfloor \sqrt{\big((b^n+1)-1\big)/2} \right\rfloor = \left\lfloor \sqrt{b^n/2} \right\rfloor$ and let $\mathcal{G}_M$ be as in Equation 8. That is, $\mathcal{G}_M$ is the set of reduced fractions x/y satisfying: $|x| \leq M$, $1 \leq |y| \leq M$, and $\gcd(b^n+1, y) = 1$. M is chosen to be much smaller than N according to eq. (9) and eq. (14). We define encoding as follows:

ModFV.Encode. For $x/y \in \mathcal{G}_M \subseteq \mathcal{F}_N$,
output $h = \mathsf{PIE.Encode}\big(x/y\big) \in \mathbb{Z}/(b^n+1)\mathbb{Z}$.

ModFV.Decode. For $h \in \mathbb{Z}/(b^n+1)\mathbb{Z}$,
output $x/y = \mathsf{PIE.Decode}(h) \in \mathcal{F}_N$.

$b^n + 1$ prime Note that since b^n+1 is prime, the function H_{b^n+1} maps x/y to $xy^{-1} \bmod b^n+1$ (definition 5). Further, since $\gcd(y, b^n+1) = 1$ for all $0 < y \leq N$, no fractions are discarded because of the gcd condition in definition 1 - i.e. all x/y with $|x|, |y| \leq N$ can be encoded.

Choosing b and n for b^n+1 a prime As one might suspect, there are rather few choices for b and n which make b^n+1 prime. The known *Fermat primes*[4] are too small for the parameter requirements of ModFV. In our search for suitable primes, we found OEIS sequence A056993 which lists primes of the form $k^{2^n}+1$. While this sequence does not provide many candidates, this is not a problem since b and n are public parameters. In particular, one can reuse an appropriately-chosen prime b^n+1 as needed without compromising security.

b^n+1 composite For a composite b^n+1, the mapping H_{b^n+1} is defined by the CRT, which requires (non-trivial) co-prime factors of b^n+1 to be known. This could be problematic, as $n \geq 1024$ will make b^n+1 very large even for small b. The following lemma addresses this difficulty.

Proposition 9. *If g is a positive integer and $x/y \in \mathcal{F}_N$, then $H_g(x/y) = xy^{-1} \bmod g$.*

Proof. This is immediate if g is prime, so suppose g is composite with prime factorization $g = p_1^{r_1} \cdots p_k^{r_k}$. Let $x/y \in \mathcal{F}_N$, $h_i = H_{p_i^{r_i}}(x/y)$, and $h = H_g(x/y)$. By definition 5,

$$h = \mathsf{CRT}_{p_1^{r_1},\ldots,p_k^{r_k}}(h_1, \ldots, h_k)\,.$$

By the definition of the CRT, h is the unique integer in $\mathcal{Z}_g$ such that $h = h_i \bmod p_i^{r_i}$. Put $h' = xy^{-1} \bmod g$, so $yh' = x \bmod g$. Since each $p_i^{r_i}$ divides g, $yh' = x \bmod p_i^{r_i}$. Multiply both sides of the preceding equation by the inverse of y modulo $p_i^{r_i}$ to get $h' = xy^{-1} \bmod p_i^{r_i}$. But this means that $h' = h_i \bmod p_i^{r_i}$, whence $h' = h$. This completes the proof.

Choosing b and n for b^n+1 composite Since we encode with H_{b^n+1}, b must be chosen carefully to ensure the message space $\mathcal{G}_M$ contains the desired fractions. For example, if we want to encode $1/2$, then we choose b a multiple of 2, whence $\gcd(2, b^n+1) = 1$ and $H_{b^n+1}(1/2)$ is defined. n may be chosen independently of b according to requirments for ModFV.

As noted above, b^n+1 may be large enough to make factoring infeasible. In this case, determining the entire input space is also infeasible, because one must enforce the condition: $\gcd(y, b^n+1) \neq 1 \implies x/y \notin \mathcal{F}_N$. This is not a problem however, as we only need a suitable *subset* of $\mathcal{F}_N$; namely $\mathcal{G}_M$. We note that if y and b have the same prime factors, then $\gcd(y, b^n+1) = 1$, whence we can encode x/y as long as every prime factor of y is a factor of b. For example, we may choose $b = p_1 p_2 \cdots p_k$, the product of the first k primes for some $k \geq 1$, meaning we can encode all $x/y \in \mathcal{G}_M$ such that any prime factor of y is one of $p_1, \ldots, p_k$. This approach can certainly give us a sufficiently large set of fractions as the message space of PIE, though this set may not be the entirety of $\mathcal{G}_M$.

We further distinguish the case where $b = p$ is prime, for this allows us to encode certain p-adic *non*-integers (p-adic numbers with negative valuation). In particular, since p and p^n+1 are always co-prime, we can encode rationals of the form x/p^k ($k > 0$) that are contained in $\mathcal{F}_N$.

[4] Primes of the form $2^{2^n}+1$

Compatible Circuits In [4] the performance of ModFV is assessed by evaluating so-called *regular (arithmetic) circuits*. We directly apply the bounds from their analysis on such circuits to our encoder to FV. A regular circuit is parameterized by non-negative integers A, D, L, and consists of evaluating A levels of additions followed by one level of multiplication, iterated D times, where inputs are integers from $[-L, L]$. Note that such a circuit has multiplicative depth D. It was shown in [10] that the output c of a regular circuit C satisfies:

$$|c| \le V(A, D, L) = L^{2D} 2^{2A(2^D-1)} \tag{13}$$

We define permitted circuits in essentially the same way as Section 4.1.

Definition 8. *For fixed A, D, L, an arithmetic circuit C is a (A, D, L)*-permitted circuit *if every input being bounded in absolute value by L implies the output is bounded in absolute value by $V(A, D, L)$.*

Equation 13 implies every regular circuit parameterized by A, D, L is an (A, D, L)-permitted circuit. When the context is clear, we will omit "(A, D, L)" simply write "permitted circuit".

Lemma 2. *Fix non-negative integers A, D, L. Let C be an arithmetic circuit, f be the multivariate polynomial that C computes, $|f|_1$ be the ℓ_1 norm of f, and $V = V(A, D, L)$. If $|f|_1 L^{\deg(f)} < V$ or equivalently,*

$$\deg(f) < 2D + \frac{2A(2^D-1) - \log(|f|_1)}{\log(L)} \tag{14}$$

then C is a permitted circuit.

Proof. Let C be an arithmetic circuit, and f be the k-variate polynomial which C computes. We can express f in the form $\sum_{i=1}^{I} c_i f_i$, where the f_i are monomials and the c_i are the coefficients.

For $\mathbf{x} \in [-L, L]^k$ and $\mathbf{L} = (L, L, \ldots, L) \in \{L\}^k$, we use the triangle inequality and $\deg(f_i) \le \deg(f)$ to obtain

$$|f(\mathbf{x})| = \left|\sum_{i=1}^{I} c_i f_i(\mathbf{x})\right| \le \left|\sum_{i=1}^{I} c_i f_i(\mathbf{L})\right| \le \left|\sum_{i=1}^{I} c_i L^{\deg(f)}\right| \le |f|_1 L^{\deg(f)}$$

The above inequalities yield $|f(\mathbf{x})| \le V$, completing the proof.

To guarantee that PIE works seamlessly with ModFV, we must ensure that the maximum degree of polynomials compatible with ModFV does not exceed the maximum degree of polynomials compatible with PIE. Thus, according to lemma 2 and equation 9, we require

$$\frac{\log(V) - \log(|f|_1)}{\log(L)} < \frac{\log(N) - \log(t)}{t \log(M)},$$

where f computes an (A, D, L)-permitted circuit, and $\mathcal{P}_{d,t}$ is the set of polynomials with which PIE is compatible. In practice, this inequality is easily satisfied because $\log(N)/\log(M)$ is quite large and t is chosen to be small.

5.2 PIE vs. CLPX: Input Space Advantage

Chen et al. ([4]) adapt the polynomial encoding idea from previous works while addressing the problem of plaintext polynomial coefficient growth. As explained above, to obtain the maximum circuit depth (corresponding to homomorphic computation) for PIE with ModFV we can directly use their analysis. Table 2 shows that when used with PIE scheme, the multiplicative depths of circuits compatible with ModFV are almost same as when used with CLPX encoding.

Number of additions $A = 0$

		$L = 2^8$		$L = 2^{16}$		$L = 2^{32}$		$L = 2^{64}$		
n	$\log_2 q$	b	$\max D$	b	$\max D$	b	$\max D$	b	$\max D$	
2^{14}	435	257	14	257	13	257	12	257	11	[4]
2^{15}	890	2^{16}	16	2^{16}	15	2^{32}	15	2^{32}	14	
2^{14}	435	2^{16}	11	2^{16}	11	2^{32}	11	2^{32}	11	Our work
2^{15}	890	2^{16}	15	2^{16}	14	2^{32}	14	2^{32}	13	

Number of additions $A = 3$

		$L = 2^8$		$L = 2^{16}$		$L = 2^{32}$		$L = 2^{64}$		
n	$\log_2 q$	b	$\max D$	b	$\max D$	b	$\max D$	b	$\max D$	
2^{14}	435	128	13	2^{11}	13	724	12	431	11	[4]
2^{15}	890	2^{28}	16	2^{22}	15	2^{19}	14	2^{35}	14	
2^{14}	435	2^{16}	10	2^{16}	10	2^{16}	10	2^{16}	10	Our work
2^{15}	890	2^{16}	15	2^{16}	14	2^{32}	14	2^{32}	13	

Table 2: Comparison of maximum circuit depth D with ModFV and PIE+ModFV.

The definition of the CLPX input space $\mathcal{P}$ depends on whether $b \geq 2$ is even or odd. If b is odd, then $b^n + 1$ is even, which means no fractions with even denominators can be encoded, and, moreover, $b^n + 1$ will not be prime. We consider the odd case to be too restrictive, and, therefore, only compare the input space of PIE with the input space of CLPX when b is even.

Proposition 10. *For b even, the cardinality of the input space $\mathcal{P}$ is $\frac{b^n-1}{b-1}$.*

By proposition 5 and eq. (7), when $b^n + 1$ is prime[5], the cardinality of $\mathcal{F}_N$ is approximately $0.6(b^n + 1)$. Consequently, using proposition 10, we see the cardinality of $\mathcal{F}_N$ is roughly $0.6(b - 1)$-times[6] the size of $\mathcal{P}$. Thus our input space is larger when $b \geq 3$, and our size advantage is directly proportional to the size of b, as shown in table 3.

[5] Primes of the form "$b^n + 1$" chosen from *https://oeis.org/A056993*.

[6] Since b^n is quite large, $\frac{|\mathcal{F}_n|}{|\mathcal{P}|} \approx \frac{0.6(b^n+1)(b-1)}{b^n-1} \approx 0.6(b-1)$

b	150	824	1534
n	2^{11}	2^{10}	2^{12}
PIE ($\lvert\mathcal{F}_N\rvert$)	$0.6(150^{2^{11}}+1)$	$0.6(824^{2^{10}}+1)$	$0.6(1534^{2^{12}}+1)$
CLPX ($\lvert\mathcal{P}\rvert$)	$\frac{150^{2^{11}}-1}{149}$	$\frac{824^{2^{10}}-1}{823}$	$\frac{1534^{2^{12}}-1}{1533}$
$\frac{\text{PIE}}{\text{CLPX}}$	86	600	857

Table 3: Comparison of input space sizes for PIE and CLPX when b^n+1 is prime. The values of n are chosen according to the security recommendations for FV.

b	3	5	7	6	30	30	210	210
n	12	8	8	16	4	8	4	6
PIE	442765	324646	4787969	$\approx 1.7\times10^{12}$	≈ 487992	$\approx 4\times10^{11}$	$\approx 1.2\times10^{9}$	$\approx 4.4\times10^{13}$
CLPX	265720	97656	960800	$\approx 5.6\times10^{11}$	27931	$\approx 2.2\times10^{10}$	$\approx 9\times10^{6}$	$\approx 4.1\times10^{11}$
$\frac{\text{PIE}}{\text{CLPX}}$	1.7	3.3	5	3	16.7	16.7	125	111.1

Table 4: Comparison of input space sizes $\mathcal{F}_N$ (for PIE) and $\mathcal{P}$ (for CLPX) when b^n+1 is composite.

For b^n+1 composite, our size advantage seems to remain, though it is less clear-cut than the prime case, since our examples use quite small b and n.
In table 4, we estimate the size of $\mathcal{F}_N$ by using proposition 5 and the approximation $\Phi(n)\approx 3n^2/\pi^2$. Note that, in practice, the size of b and n will be much larger than the numbers provided in the table, and we cannot speculate to how the relationship between $|\mathcal{F}_N|$ and $|\mathcal{P}|$ varies as b and n become large enough for practical applications.

6 Experimental Results

We implemented PIE (in C++) together with proof-of-concept implementations of IDGHV and ModFV schemes[7] using NTL [?].

Since our encoding does not affect the run time of the underlying HE scheme we provide benchmark times taken for encoding and decoding only. We estimated the runtime of encoding and decoding using two sets, each containing 10,000 rational numbers. The first set contains rationals with numerator and denominator up to 32 bits and the second set contains rationals with numerator and denominator up to 64 bits. These sets are simply the message space

[7] FHE part of our implementation is not optimized

$\mathcal{G}_M = \{x/y \big| |x| \leq M, 0 < y \leq M\}$ for $M = 2^{32} - 1$ and $M = 2^{64} - 1$, respectively. Runtimes are obtained as the average runtime over all the elements in each set. The results are shown in table 5. All experiments are done on a MacBook Pro with Apple M1 Max, 32 GB RAM, 1TB SSD. Our implementation can be found at `https://github.com/Algemetric/pie-cpp`.

$\lvert p\rvert_{\text{bits}}$	650	650	1250	1250	3200	3200
$\lvert M\rvert_{\text{bits}}$	32	64	32	64	32	64
Encode time	0.023833 ms	0.001958 ms	0.006584 ms	0.001708 ms	0.003916 ms	0.002375 ms
Decode time	0.047792 ms	0.054791 ms	0.028625 ms	0.0475 ms	0.046625 ms	0.06175 ms

Table 5: Average encoding and decoding times for various parameters. Here p is the prime used for encoding and decoding.

Our implementation of encoding and decoding is not optmized for performance. We have used NTL for computing inverse in the encoding function. For the MEEA in decoding, we implemented the (truncated) extended Euclidean algorithm.

Acknowledgements

We thank Jonathan Katz for helpful discussions. This work is fully supported by Algemetric.

References

1. Arita, S., Nakasato, S.: Fully homomorphic encryption for point numbers. Cryptology ePrint Archive, Report 2016/402 (2016), `https://ia.cr/2016/402`
2. Bonte, C., Bootland, C., Bos, J.W., Castryck, W., Iliashenko, I., Vercauteren, F.: Faster homomorphic function evaluation using non-integral base encoding. In: Fischer, W., Homma, N. (eds.) CHES 2017. LNCS, vol. 10529, pp. 579–600. Springer, Heidelberg (Sep 2017). https://doi.org/10.1007/978-3-319-66787-4˙28
3. Bos, J.W., Lauter, K.E., Naehrig, M.: Private predictive analysis on encrypted medical data. Journal of biomedical informatics **50**, 234–43 (2014)
4. Chen, H., Laine, K., Player, R., Xia, Y.: High-precision arithmetic in homomorphic encryption. In: Cryptographers' Track at the RSA Conference. pp. 116–136. Springer (2018)
5. Cheon, J.H., Coron, J.S., Kim, J., Lee, M.S., Lepoint, T., Tibouchi, M., Yun, A.: Batch fully homomorphic encryption over the integers. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 315–335. Springer (2013)

6. Cheon, J.H., Jeong, J., Lee, J., Lee, K.: Privacy-preserving computations of predictive medical models with minimax approximation and non-adjacent form. In: Brenner, M., Rohloff, K., Bonneau, J., Miller, A., Ryan, P.Y.A., Teague, V., Bracciali, A., Sala, M., Pintore, F., Jakobsson, M. (eds.) FC 2017 Workshops. LNCS, vol. 10323, pp. 53–74. Springer, Heidelberg (Apr 2017)
7. Cheon, J.H., Kim, A., Kim, M., Song, Y.S.: Homomorphic encryption for arithmetic of approximate numbers. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part I. LNCS, vol. 10624, pp. 409–437. Springer, Heidelberg (Dec 2017). https://doi.org/10.1007/978-3-319-70694-8˙15
8. Cheon, J.H., Kim, J., Lee, M.S., Yun, A.: Crt-based fully homomorphic encryption over the integers. Information Sciences **310**, 149–162 (2015). https://doi.org/https://doi.org/10.1016/j.ins.2015.03.019, https://www.sciencedirect.com/science/article/pii/S002002551500184X
9. Costache, A., Smart, N.P., Vivek, S., Waller, A.: Fixed point arithmetic in she scheme. Cryptology ePrint Archive, Paper 2016/250 (2016), https://eprint.iacr.org/2016/250, https://eprint.iacr.org/2016/250
10. Costache, A., Smart, N., Vivek, S., Waller, A.: Fixed point arithmetic in SHE scheme. Cryptology ePrint Archive, Report 2016/250 (2016), https://eprint.iacr.org/2016/250
11. Costache, A., Smart, N.P.: Which ring based somewhat homomorphic encryption scheme is best? In: Sako, K. (ed.) CT-RSA 2016. LNCS, vol. 9610, pp. 325–340. Springer, Heidelberg (Feb / Mar 2016). https://doi.org/10.1007/978-3-319-29485-8˙19
12. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. Cryptology ePrint Archive, Paper 2009/616 (2009), https://eprint.iacr.org/2009/616, https://eprint.iacr.org/2009/616
13. Dowlin, N., Gilad-Bachrach, R., Laine, K., Lauter, K., Naehrig, M., Wernsing, J.: Manual for using homomorphic encryption for bioinformatics. Proceedings of the IEEE **105**(3), 552–567 (2017). https://doi.org/10.1109/JPROC.2016.2622218
14. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. IACR Cryptol. ePrint Arch. **2012**, 144 (2012)
15. Gregory, R.: Error-free computation with rational numbers. BIT Numerical Mathematics **21**(2), 194–202 (1981)
16. Hoffstein, J., Silverman, J.: Optimizations for ntru. public-key cryptography and computational number theory (2002)
17. Jäschke, A., Armknecht, F.: Accelerating homomorphic computations on rational numbers. In: Manulis, M., Sadeghi, A.R., Schneider, S. (eds.) ACNS 16. LNCS, vol. 9696, pp. 405–423. Springer, Heidelberg (Jun 2016). https://doi.org/10.1007/978-3-319-39555-5˙22
18. Knuth, D.E.: Art of computer programming, volume 2: Seminumerical algorithms. Addison-Wesley Professional (2014)
19. Koç, Ç.K.: Parallel p-adic method for solving linear systems of equations. Parallel Computing **23**(13), 2067–2074 (1997)
20. Krishnamurthy, E.V.: Error-free polynomial matrix computations. Springer Science & Business Media (2012)
21. Lauter, K.E., López-Alt, A., Naehrig, M.: Private computation on encrypted genomic data. In: Aranha, D.F., Menezes, A. (eds.) LATINCRYPT 2014. LNCS, vol. 8895, pp. 3–27. Springer, Heidelberg (Sep 2015). https://doi.org/10.1007/978-3-319-16295-9˙1
22. Li, X., Lu, C., Sjogren, J.A.: A method for Hensel code overflow detection. ACM SIGAPP Applied Computing Review **12**(1), 6–11 (2012)

23. Lu, C., Li, X.: An introduction of multiple p-adic data type and its parallel implementation. In: 2014 IEEE/ACIS 13th International Conference on Computer and Information Science (ICIS). pp. 303–308. IEEE (2014)
24. Mahler, K.: Introduction to p-adic numbers and their functions. No. 64, CUP Archive (1973)
25. Mahler, K., et al.: Part 1: p-adic and g-adic numbers, and their approximations. In: Lectures on Diophantine Approximations, pp. 1–2. University of Notre Dame (1961)
26. Mukhopadhyay, A.: A solution to the polynomial Hensel-code conversion problem. In: European Conference on Computer Algebra. pp. 327–327. Springer (1985)
27. Naehrig, M., Lauter, K., Vaikuntanathan, V.: Can homomorphic encryption be practical? In: Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop. p. 113–124. CCSW '11, Association for Computing Machinery, New York, NY, USA (2011). https://doi.org/10.1145/2046660.2046682, https://doi.org/10.1145/2046660.2046682
28. Rao, T.M., Gregory, R.T.: The conversion of hensel codes to rational numbers. In: 1981 IEEE 5th Symposium on Computer Arithmetic (ARITH). pp. 10–20. IEEE (1981)
29. Shoup, V.: A Computational Introduction to Number Theory and Algebra. Cambridge University Press, USA, 2 edn. (2009)

A Appendix: Encodings with Primes and Prime Powers

Assume we want to encode the following fractions:

$$m_1 = -\frac{2}{3}, m_2 = -\frac{1}{2}, m_3 = \frac{1}{3}. \tag{15}$$

Let $p = 11$ and $r = 3$, so $p^r = 1331$ and $N = \lfloor\sqrt{(p^r - 1)/2}\rfloor = 25$. Since the above fractions lie in $\mathcal{F}_{25}$, we can encode them as follows:

$$m_1 = H_{1331}\left(-\tfrac{2}{3}\right) = 443,$$

$$m_2 = H_{1331}\left(-\tfrac{1}{2}\right) = 665,$$

$$m_3 = H_{1331}\left(\tfrac{1}{3}\right) = 444.$$

Due to the restriction $\gcd(\text{denominator}, p^r) = 1$, many fractions x/y which satisfy $|x|, |y| \leq N$ cannot be encoded. E.g., when $p^r = 11^3$, $23/22$ cannot be encoded. Of course, this is because the mapping H_{p^r} requires the inverse of the denominator modulo p^r, which does not exist when $\gcd(\text{denominator}, p^r) \neq 1$.

A.1 Choosing the Encoding Parameters p and r

Let $\mathcal{S}$ be a set of fractions such that

$$\mathcal{S} = \left\{-\frac{13}{25}, \frac{23}{19}, \frac{31}{5}, \frac{17}{61}, \frac{48}{23}\right\}.$$

One can choose a prime that is sufficient for encoding and decoding all fractions by simply checking the largest numerator or denominator in absolute value and set it as the value of b and then find the right prime p such that

$$p \geq 2b^2 + 1.$$

The largest quantity in $\mathcal{S}$ is 61, so we set $b = 61$ which means we need a prime p that satisfies

$$p \geq 7443.$$

The smallest prime to satisfy the above inequality is 7451 which gives $N = \left\lfloor \sqrt{(7451-1)/2} \right\rfloor = 61$. That allows us to encode all fractions in $\mathcal{S}$. We emphasize that this process works for *any* finite set of rationals.
Equivalently, one could choose a small prime which is co-prime with all of the denominators, and then choose an exponent r large enough to allow the fractions to be encoded. For example, $p = 3$ is co-prime with all denominators in $\mathcal{S}$, which means we must choose r large enough so that $3^r \geq 2(61)^2 + 1 = 7443$. That is,

$$r \geq \frac{\log(7443)}{\log(3)} \approx 8.1.$$

So $p^r = 3^9$ also suffices to encode the members of $\mathcal{S}$.

However, can we actually do something with it? If we hope to compute over the image of $\mathcal{S}$, we need to choose a prime (power) that allows "room" for including the outputs of the operations we expect to work with. Instead of choosing a prime from strict parameters, a more conservative approach could be to consider the bit length of the largest numerator or denominator and the function one wishes to compute. If this time we let b be the bit-length of the largest numerator or denominator in absolute value and the function be $f(x_1, x_2, \ldots, x_n) = x_1 x_2 \cdots x_n$, then we need a prime that satisfies the following inequality:

$$|p|_{\text{bits}} > 2bn + 1.$$

Say that we have $n = 5$. Since 61 is a 6-bit number, we set $b = 6$. We now need a prime such that

$$|p|_{\text{bits}} > 61.$$

We choose $p = 3693628617552068003$, a 62-bit prime which give us the following encodings of the members of $\mathcal{S}$:

$$\begin{aligned}
h_1 &= H_p\left(-\tfrac{13}{25}\right) = 3102648038743737122,\\
h_2 &= H_p\left(\tfrac{23}{19}\right) \quad = 2138416568056460424,\\
h_3 &= H_p\left(\tfrac{31}{5}\right) \quad = 2216177170531240808,\\
h_4 &= H_p\left(\tfrac{17}{61}\right) \quad = 3390872173490423085,\\
h_5 &= H_p\left(\tfrac{48}{23}\right) \quad = 321185097178440698,
\end{aligned}$$

and we can check that

$$\prod_{i=1}^{5} h_i \bmod p = 2444130464540096986$$

which decodes to

$$H_p^{-1}(2444130464540096986) = \frac{-328848}{144875}$$

and matches

$$-\frac{13}{25} \cdot \frac{23}{19} \cdot \frac{31}{5} \cdot \frac{17}{61} \cdot \frac{48}{23} = \frac{-328848}{144875}.$$

This example shows the intuition behind Proposition 7 and Theorem 8.

B Appendix: Extending Farey Rationals for Larger Input Space

Extending the set $\mathcal{F}_\mathbf{N}$. While the Farey rationals $\mathcal{F}_N$ have a very simple description and are easy to work with, they have a downside: their size. For example, if $p = 907$, then $N = 21$ and the cardinality of $\mathcal{F}_N$ is 559. This means that $907 - 559 = 348$ integers in $\mathcal{Z}_{907}$ do not have a pre-image (under H_{907}^{-1}) in $\mathcal{F}_N$. We address this by extending $\mathcal{F}_N$ to a set $\mathcal{F}_{N,g}$

Definition 9 (Extended Farey Rationals). *For a positive integer g, the extended Farey rationals are defined as the set of reduced fractions:*

$$\mathcal{F}_{N,g} = \left\{ \frac{x}{y} \,\middle|\, \exists h \in \mathcal{Z}_g \ s.t.\ \mathsf{MEEA}(g,h) = (x,y),\ \gcd(g,y) = 1 \right\}.$$

Clearly $\mathcal{F}_N \subseteq \mathcal{F}_{N,g}$. We also note that for all $m \in \mathcal{F}_{N,g}$, $H_g^{-1}\big(H_g(m)\big) = m$ $\big($generalize proof of Proposition 1(i)$\big)$. The following lemma provides a necessary, though not sufficient, condition for a rational number to be in $\mathcal{F}_{N,g}$.

Proposition 11. *Let g be a positive integer, and $N = \left\lfloor \sqrt{(g-1)/2} \right\rfloor$. If $x/y \in \mathcal{F}_{N,g}$, then $|x| \le N$ and $|y| \le 2N+1$.*

Proof. Let $h \in \mathcal{Z}_g$, and suppose $H_g^{-1}(h) = x/y$. By definition of MEEA, $x/y = x_i/y_i$ for some x_i, y_i computed by the EEA. That $|x| \le N$ is immediate from the definition of H_g^{-1} (i.e. the stopping condition in MEEA). The outputs of the EEA satisfy [29, Theorem 4.3(v)]

$$|y_k| \le \frac{x_0}{x_{k-1}}, \ \text{for all } k.$$

By definition, $x_{i-1} > N$. Whence, for $N' = \sqrt{(g-1)/2}$,

$$|y_i| \le \frac{g}{x_{i-1}} < \frac{g}{N'} < \frac{2(N')^2+1}{N'} = 2N' + \frac{1}{N'}$$

It follows that $|y_i| \le \lfloor 2N' + 1/N' \rfloor \le 2N+1$, completing the proof.

This proposition simplifies the process of deciding whether a given reduced rational number x/y is in $\mathcal{F}_{N,g}$:

(i) If $|x| \leq N$, $|y| \leq N$, and $\gcd(g, y) = 1$, then $x/y \in \mathcal{F}_N \subset \mathcal{F}_{N,g}$.

(ii) If $|x| > N$ or $|y| > 2N + 1$ or $\gcd(g, y) > 1$, then $x/y \notin \mathcal{F}_{N,g}$.

(iii) If $|x| \leq N$, $N < |y| \leq 2N + 1$, and $\gcd(g, y) = 1$, then $x/y \in \mathcal{F}_{N,g}$ if and only if $H_g^{-1}\big(H_g(x/y)\big) = x/y$.

Two Options for the Message Space. For a fixed positive integer g, we now have two sets of rationals which can serve as the domain of the encoder:

- the Farey rationals $\mathcal{F}_N$, and
- the extended Farey rationals $\mathcal{F}_{N,g}$.

The advantage of $\mathcal{F}_N$ is its simplicity. $\mathcal{F}_{N,g}$, on the other hand, is larger than $\mathcal{F}_N$ and, when g is prime, has exactly g elements.