# PIE: $p$-adic Encoding for High-Precision Arithmetic using Homomorphic Encryption

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

[1]Algemetric

## Motivation

Rational data are frequently used in many real-world settings such as finance, health care, and business intelligence. Further, these data must often be composed in ways which require addition and multiplication of rational numbers.

## $p$-adic numbers, plaintexts, and the encoding function

**$p$-adic representation:** 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 $0 \le a_j < p$ and $n \in \mathbb{Z}$. An $r$-segment $p$-adic representation, a.k.a Hensel code, simply truncates the above sum after $j = r-1$. I.e.,

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

A $p$-adic *integer* is a *rational* which can be rewritten in the form

$$\frac{x'}{y'} p^v, \text{ for } v \ge 0 \text{ and } \gcd(x', p) = \gcd(y', p) = 1$$

Given a prime $p$ and an integer $r \ge 1$, let $N = \left\lfloor \sqrt{\frac{p^r - 1}{2}} \right\rfloor$. The Farey rationals are defined as

$$\mathcal{F}_N = \left\{ \frac{x}{y} : 0 \le |x| \le N, 1 \le y \le N, \gcd(x, y) = \gcd(y, p) = 1 \right\} \tag{2}$$

$\mathcal{F}_N$ is a set of $p$-adic integers.

Given $x_0, x_1 \in \mathbb{Z}$,

$$\mathsf{MEEA}(x_0, x_1) \longrightarrow \left((-1)^{i+1} x_i,\ (-1)^{i+1} y_i\right), \quad i \ge 0 \tag{3}$$

where $x_i, y_i$ are generated by the extended Euclidean algorithm (**EEA**). The **MEEA** simply stops **EEA** early (once $|x_i| \le N$).

**A homomorphic mapping:** The *injective* mapping $H_{p^r} : \mathcal{F}_N \to \mathbb{Z}/p^r\mathbb{Z}$ and its inverse are defined as

$$H_{p^r}\left(\frac{x}{y}\right) = x y^{-1} \bmod p^r, \tag{4}$$

$$H^{-1}_{p^r}(h) = \mathsf{MEEA}(p^r, h) \tag{5}$$

$H_{p^r}$ gives a unique representation of each element of $\mathcal{F}_N$ in $\mathbb{Z}/p^r\mathbb{Z}$.

**General extension:** The above results can be extended when $p^r$ is replaced by an arbitrary positive integer $g = p_1^{r_1} \cdots p_k^{r_k}$ to define $H_g : \mathcal{F}_{N,g} \to \mathbb{Z}/g\mathbb{Z}$,

where the domain is the set of *extended* Farey rationals:

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

## PIE Encoder

Let $g$ be a positive integer and $N = \lfloor \sqrt{(g-1)/2} \rfloor$. $\mathcal{F}_{N,g}$ is the input space. Since it is not closed under $+$ and $\cdot$, we define $\mathcal{G}_M = \left\{ x/y \,\middle|\, |x|, |y| \le M \right\} \subseteq \mathcal{F}_{N,g}$ as the message space.

Encoding and decoding are defined as follows:

$$\mathsf{PIE.Encode}\left(\frac{x}{y}\right). \text{ For } \frac{x}{y} \in \mathcal{F}_{N,g} \text{ output } H_g\left(\frac{x}{y}\right) \in \mathbb{Z}/g\mathbb{Z}. \tag{6}$$

$$\mathsf{PIE.Decode}(z). \text{ For } z \in \mathbb{Z}/g\mathbb{Z}, \text{ output } H_g^{-1}(z) \in \mathcal{F}_{N,g}. \tag{7}$$

The smaller $M$ is relative to $N$, the deeper the circuits with which **PIE** is compatible.

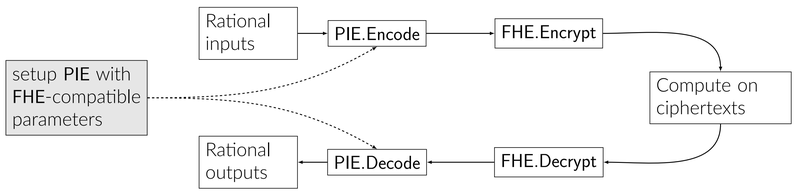


Figure 1. Attaching **PIE** to an **FHE** scheme.

## PIE with an AGCD-based Batch FHE

We attach **PIE** to the batch integer FHE scheme (**IDGHV**) from [2].

Choose the public parameters $Q_1, \dots, Q_\ell$ of **IDGHV** to be distinct odd primes, let $g = \prod_{i=1}^{\ell} Q_i$, and $N = \lfloor \sqrt{(g-1)/2} \rfloor$. We encode and decode as follows:

**IDGHV.Encode.** For $m \in \mathcal{G}_M$, output $\left(\mathsf{PIE.Encode}(m) \bmod Q_1, \dots, \mathsf{PIE.Encode}(m) \bmod Q_\ell\right)$

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

The set $\mathcal{P}_{d,t}$ is the set of multivariate polynomials with integer coefficients, degree $d$, and $\ell_1$ norm at most $t$. Encoding correctness, i.e. $\mathsf{Decode}\big(\mathsf{Encode}(m)\big) = m$, is ensured by choosing $M \le \left(N/t\right)^{1/dt}$.

## PIE with an RLWE-based FHE

We attach **PIE** to a modified version [1] of the Fan-Vercauteren (**FV**) scheme [3], which we call **ModFV**. The message space of **ModFV** is the ring $\mathbb{Z}/(b^n+1)\mathbb{Z}$, for public HE parameters $b$ and $n$. We distinguish (for analysis) two cases: $b$ *prime* and $b$ *composite*. The choice of $b$ decides the set of rationals which can be encoded.

**ModFV.Encode.** For $x/y \in \mathcal{G}_M \subseteq \mathcal{F}_{N,b^n+1}$, output $\mathsf{PIE.Encode}\left(x/y\right) \in \mathbb{Z}/(b^n+1)\mathbb{Z}$.

**ModFV.Decode.** For $h \in \mathbb{Z}/(b^n+1)\mathbb{Z}$, output $\mathsf{PIE.Decode}(h)$.

**ModFV** is also paired with a rational encoder, however, for fixed $b$,$n$ **PIE** has a much larger input space.

| $b$ | $n$ | PIE | CLPX | PIE/ModFV |
|---|---|---|---|---|
| 150 | $2^{11}$ | $150^{2^{11}} + 1$ | $\frac{150^{2^{11}}-1}{149}$ | 143 |
| 824 | $2^{10}$ | $824^{2^{10}} + 1$ | $\frac{824^{2^{10}}-1}{823}$ | 1000 |
| 1534 | $2^{12}$ | $1534^{2^{12}} + 1$ | $\frac{1534^{2^{12}}-1}{1533}$ | 1429 |

Table 1. Comparison of input space sizes for **PIE** and **ModFV** encoder when $b^n + 1$ is prime.

| $b$ | $n$ | PIE | ModFV | PIE/ModFV |
|---|---|---|---|---|
| 3 | 12 | 442765 | 265720 | 1.7 |
| 5 | 8 | 324646 | 97656 | 3.3 |
| 7 | 8 | 4787969 | 960800 | 5 |
| 30 | 8 | $\approx 4 \times 10^{11}$ | $\approx 2.2 \times 10^{10}$ | 16.7 |
| 210 | 4 | $\approx 1.2 \times 10^{9}$ | $\approx 9 \times 10^{6}$ | 125 |
| 210 | 6 | $\approx 4.4 \times 10^{13}$ | $\approx 4.1 \times 10^{11}$ | 111 |

Table 2. Comparison of input space sizes for **PIE** and **ModFV** encoder when $b^n + 1$ is composite.

## Library and Demo

**PIE** is implemented in C++ using NTL and GMP for large integer, vector, and polynomial arithmetic, and Bazel as the build system, and is available at https://github.com/Algemetric/pie-cpp. **PIE** can be built as a stand-alone fractional encoding library or in conjunction with either **IDGHV** or **ModFV**.

The implementation is currently not optimized for performance, therefore the numbers below aim to highlight the overhead of adding **PIE** to an HE scheme.

| $\lambda$ | $\ell$ | $\rho$ | $\eta$ | $\gamma$ | $\tau$ | $\log_2$(prime) | Encoding time |
|---|---|---|---|---|---|---|---|
| 50 | 6 | 60 | 4248 | $5.3 \cdot 10^8$ | 661 | 1500 | 0.006ms |
| 52 | 37 | 41 | 1558 | $9 \cdot 10^5$ | 661 | 358 | 0.007833ms |

Table 3. Example of encoding times (for IDGHV scheme).

**PIE** is also implemented and available at http://api.inkasso.obscura.algemetric.com/login, a web application for insight generation and visualization over encrypted data. For encoding purposes, any number at least 128 bits in length is enough to encode a data set with more than 4 million records. Moreover, for this type of application, 128-bit primes suffice for encoding. However, the actual sizes of the primes depend on the underlying HE scheme.

## References

[1] H. Chen, K. Laine, R. Player, and Y. Xia. High-precision arithmetic in homomorphic encryption. In *CT-RSA*. Springer, 2018.

[2] J. H. Cheon, J.-S. Coron, J. Kim, M. S. Lee, T. Lepoint, M. Tibouchi, and A. Yun. Batch fully homomorphic encryption over the integers. In *Eurocrypt*. Springer, 2013.

[3] J. Fan and F. Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptol. ePrint Arch.*, 2012.