An Efficient Homomorphic Data Encoding with Multiple Secret Hensel Codes

David W. H. A. da Silva, Carlos Paz de Araujo, and Edward Chow

***Abstract*—Data encoding is widely used for a variety of reasons. Encoding schemes in general serve to convert one form of data to another in order to enhance the efficiency of data storage, transmission, computation and privacy, to name just a few. When it comes to privacy, data may be encoded to hide its meaning from direct access or encrypted to attain a certain security level. If the encoding scheme preserves additive and multiplicative homomorphisms, then operations on encoded data may be performed without prior decoding, which improves the utility of such mechanism. We introduce a probabilistic fully homomorphic encoding scheme that is practical as a stand-alone entry-level solution to data privacy or as an added component of existing encryption schemes, especially those that are deterministic. We demonstrate how the finite segment of p-adic numbers can be explored to derive probabilistic multiple secret Hensel codes which yields multiple layers of obscurity in an efficient way. Our encoding scheme is compact, ultra lightweight and suitable for applications ranging from edge to cloud computing. Without significant changes in its mathematical foundation, as a proposed continuation of this present work, further investigation can take place in order to confirm if the same encoding scheme can be extended to be a standalone secure instance of a fully homomorphic encryption scheme.**

***Index Terms*—Data encoding, p-adic numbers, g-adic numbers, hensel code, secret Hensel codes, homomorphic encoding.**

I. Introduction

Among its many applications, encoding techniques facilitate certain procedures with data such as storage and traffic [1], recovery [2], compression [3]-[5], signal processing [6], to cite a few. Classical examples of data encoding techniques include JPEG [7], MPEG [8], ASCII [9] and UTF-8 [10]. In this work we examine data encoding as it relates to privacy. With the advent of newer technologies, such as IoT and 5G wireless protocols, the volume of stored and transmitted data is increasing exponentially. This phenomenon affects not only their supporting infrastructure [11] but also raises concerns about data privacy. Data encoding has also been used to improve privacy through data hiding techniques [12] such as steganographic encoding for digital images [13], [14], secret error-correcting codes [15], audio data hiding [16], grammar encoding [17], and DNA encoding [18], [19]. It is safe to assume that data encoding has the potential to play a relevant role in every stage of the data life-cycle.

We are particularly interested in novel and efficient ways by which data can be represented while offering some level of secrecy. Our goal is to propose a flexible, efficient and general purpose data encoding scheme that preserves additive and multiplicative homomorphisms and can be either integrated into existing encryption schemes or serve as a single hiding technique for a variety of applications in Engineering and Computer Science such as digital signal processing [20], cyber physical systems [21], machine learning [22], edge computing [23], parallel processing [24], cloud applications [25], among others.

A. Our Contribution

We propose a probabilistic fully homomorphic encoding mechanism based on multiple secret Hensel codes, as an extension of the finite segment of p-adic and g-adic numbers. The conventional Hensel code is extended in the following ways:

1. The prime numbers involved in the Hensel encoding are kept private which compromises the efficiency of calculating the corresponding inverse map;
2. Multiple prime numbers are used in the g-adic setting where random numbers are considered as part of the data we want to encode, which gives the encoding scheme a probabilistic nature;
3. From any given integer to be encoded, we first force the generation of a rational number via the inverse g-adic mapping, so secret Hensel codes are generated.

This presents a challenge in the decoding mechanism if the prime numbers are not known. Among the possible applications enabled by our scheme we highlight the data encoding of existing encryption schemes, especially those that do not possess randomization in their encryption function. It may also be applied to functions over structured data such as audio, video, image, signals, and in a large variety of applications that afford a reasonable data expansion in favor of meaningful obfuscation.

B. Related Work

Although the theory of p-adic numbers were formally introduced in the end of the nineteenth century [26], Hensel codes were only proposed as a theory on its own between the late 1970s and early 1980s [27]-[33]. Since then, Hensel codes have been applied for a variaty of purposes, including error- free, parallel and distributed computation as seen in [34]-[39].

As a recent example of the advantages of applying Hensel codes in real-world solutions, Barillas demonstrated an efficient quantization method for a new classification approach in machine learning based on finite p-adic arithmetic over the MNIST dataset [40]. Barillas obtained a peformance 42 times faster and energy consumption 62 times lower than current state-of-the-art [41]. Hensel codes has also been recently used as an encoding component for

Manuscript received January 1, 2020; revised March 12, 2020.

David W. H. A. da Silva, Carlos Paz de Araujo, and Edward Chow are with are the University of Colorado at Colorado Springs, CO 80918 USA (e-mail: dhonorio@uccs.edu, cpazdear@uccs.edu, cchow@uccs.edu).

doi: 10.18178/ijiee.2025.15.3.1

experimental encryption schemes and related protocols [42]-[44].

C. Preliminaries

We denote the bit length of any number x as $|x|_{\text{bits}}$. If a random number y is said to be *uniformly generated* and we provide that $|y|_{\text{bits}} = |x|_{\text{bits}}$, then the range of y is $2^{|y|_{\text{bits}}-1}$ ... $2^{|y|_{\text{bits}}} - 1$. If we provide that $|y|_{\text{bits}} \leq |x|_{\text{bits}}$, then the range of y is given by 0 ... $2^{|y|_{\text{bits}}} - 1$. A set S is denoted as usual, that is $S = \{s_1, \ldots, s_k\}$, while an ordered list L is denoted as $L = (l_1, \ldots, l_k)$. We seek to only add notation and nomenclature in order to favor readability and implementation.

II. Hensel Codes

As an alternative for the application of infinite precision integer and rational arithmetic [45], which are generally expensive in terms of space and time consumption, Hensel proposed the p-adic number system [26] (for p as an odd prime number) with which one can perform calculations on integers as a replacement of the operations on rational numbers, which waives the need of dealing with the costs of rational arithmetic while achieving the same results. Krishnamurthy, Rao and Subramanian further developed the ideas of Hensel to create a finite segment of p-adic numbers, which they named Hensel codes, as an isomorphic map between a specialized subset of the rational numbers and integers over a finite field modulo p. Rao and Gregory [27] remarked that converting an integer back into rational number was difficult, so they proposed methods to alleviate this problem. However, it was Miola in [33] that introduced a general and efficient method for calculating the direct and inverse mappings of Hensel codes.

The reader might refer to Miola in [33] for detailed information about the unique direct and inverse maps between a certain set of rational numbers and their corresponding Hensel codes, which we hope will create confidence about the correctness, efficiency in representation and applicability of both p-adic and g-adic numbers in modern real-world applications. Here, we rearrange the theorems and proofs of interest in order to make our discussion self-contained, which we hope to help the appreciation of the fundamental notions in our contribution, in particular, how Hensel codes can be explored to provide a fully homomorphic data encoding scheme.

A. Extended Euclidean Algorithm

Due to its importance in the mappings between rational numbers and integers over a finite field within the theory of p-adic numbers, we provide a brief review on the basics of the Extended Euclidean Algorithm (EEA) [46]. The EEA guarantees that

$$\begin{aligned}&\gcd(a,b) = \gcd(r_0,r_1) = \gcd(r_1,r_2) = \\ &\gcd(r_2,r_3) = \cdots = \gcd(r_{k-2},r_{k-1}) = \\ &\gcd(r_{k-1},r_k) = r_k.\end{aligned} \tag{1}$$

which means that $\gcd(a,b)$ yields the last non-zero remainder in the sequence $r_0, r_1, \ldots, r_k$. We remark the description provided in [47], [48], where given two non-negative integers a_0 and a_1, the EEA computes x and y and $\gcd(a_0, a_1)$ so the following holds:

$$a_0 x + a_1 y = d = \gcd(a_0, a_1). \tag{2}$$

Within the computation of the EEA, the values for

$$\begin{aligned}&a_1, \ldots, a_n,\\ &x_1, \ldots, x_n,\\ &y_1, \ldots, y_n\end{aligned} \tag{3}$$

are calculated over each iteration such that

$$\begin{aligned}a_{i+1} &= a_{i-1} - q a_i, && \text{given } a_0, a_1,\\ x_{i+1} &= x_{i-1} - q x_i, && \text{for } x_0 = 1, x_1 = 0,\\ y_{i+1} &= y_{i-1} - q y_i, && \text{for } y_0 = 0, y_1 = 1,\end{aligned} \tag{4}$$

considering that

$$\begin{aligned}q &= \left\lfloor \frac{a_{i-1}}{a_i} \right\rfloor,\\ a_n &= 0\\ a_{n-1} &= \gcd(a_0, a_1).\end{aligned} \tag{5}$$

The property in Eq. 2 can be generalized to what is known as a *diophantine equation*, and it is given by

$$a_0 x + a_1 y = c, \tag{6}$$

where x and y are unknown integers and a_0, a_1 and c are given integers. The solution $\overline{x}, \overline{y}$ provided in [48] when $\gcd(a_0, a_1) \mid c$ is given by

$$\begin{aligned}\overline{x} &= m x_{n-1} + \frac{k a_1}{\gcd(a_0, a_1)},\\ \overline{y} &= m y_{n-1} + \frac{k a_0}{\gcd(a_0, a_1)},\end{aligned} \tag{7}$$

where k and m are integers and $c = m \cdot \gcd(a_0, a_1)$.

B. Hensel Code and Its Inverse

The isomorphic direct and inverse mappings between a certain set of rational numbers and integers modulo p^r is described below.

Theorem 1: Given a rational number c/d and p^r, where p is an odd prime number and r is an arbitrary positive integer such that c, d and p^r are pairwise relatively prime, the Hensel code of the rational number c/d, denoted as h, is calculated as follows:

$$h = c \cdot d^{-1} \bmod p^r, \tag{8}$$

for all c and d such that

$$|c|, |d| \leq \left\lfloor \sqrt{\frac{p^r}{2}} \right\rfloor. \tag{9}$$

doi: 10.18178/ijiee.2025.15.3.1

Proof 1: The value of h as in Eq. 8 can be obtained as a solution of

$$h = (\frac{c}{d}) \bmod p^r \tag{10}$$

which can also be written as

$$kp^r + hd = c \tag{11}$$

for an integer k. This equation is a diophantine equation solvable by the EEA with solutions as in Eq. 7. In this case we have

$$\overline{x} = cx_{n-1} + kd, \quad \overline{y} = cy_{n-1} + kp^r \tag{12}$$

and the assertion follows by recalling that $h < p^r$.

Example 1: Let $c/d = 5/3$, $p = 257$ and $r = 1$. The Hensel code h of c/d is given by:

$$\begin{aligned} h &= c \cdot d^{-1} \bmod p^r \\ &= 5 \cdot 3^{-1} \bmod 257 \\ &= 5 \cdot 86 \bmod 257 \\ &= 173 \end{aligned} \tag{13}$$

The intuition behind the Hensel code computation is rather straightforward. Consider the rational number $5/3$, this time without the reduction mod 257:

$$\tfrac{5}{3} = 5 \cdot \tfrac{1}{3} = 5 \cdot 3^{-1}. \tag{14}$$

We see that another way of representing $5/3$ is by writing the integer 5 multiplying the inverse of the integer 3. The Hensel code is the same calculation modulo p^r, which means, according to Example 1, that we need to calculate the modular multiplicative inverse of 3 with respect to the prime number 257.

Miola in [33] remarks that, if a Hensel code h is calculated for a particular rational number c/d as shown in Theorem 1, then, the EEA will be able to uniquely and correctly compute c/d from h.

Lemma 1: ([33]) If

$$|\frac{p}{q} - x| \leq \frac{1}{2q^2}, \tag{15}$$

then p/q is a convergent to x.

Proof 2: The proof for the lemma is provided by Hardy et al. in [49].

Theorem 2: ([33]) If c/d is such that

$$h = \frac{c}{d} \bmod p^r, \tag{16}$$

with $0 \leq |c| \leq N,\ 0 \leq |d| \leq N,\ N = \lfloor \sqrt{\frac{p^r}{2}} \rfloor$, then, since EEA applied to p^r and h computes the convergents of h/p^r, there exists an i such that the i-th convergent determines exactly $c/d = a_i/y_i$.

Proof 3: ([33]) The equation in Theorem 2 can be rewritten as

$$\frac{k}{d} + \frac{h}{p^r} = \frac{c}{dp^r} \tag{17}$$

for some k, and $-k/d$ is an approximation to h/p^r with an error of

$$|\frac{k}{d} + \frac{h}{p^r}| = \frac{c}{dp^r} < \frac{c}{2cd^2} \leq \frac{1}{2d^2}. \tag{18}$$

Lemma 1 is then applied to prove that $-k/d$ is a convergent of h/p^r and it can be computed by EEA (which computes all the convergents). Then value c/d can be obtained by EEA as a_i/y_i for the i such that $|y_i| \leq N$.

Example 2: Let $h = 173$, $p = 257$ and $r = 1$. In order to find the unique rational number for 173 we compute the EEA for $a_0 = 257$, $a_1 = 173$, $y_0 = 0$, $y_1 = 1$ as follows:

$$\begin{aligned} i &= 1 \\ q &= \lfloor \tfrac{a_0}{a_1} \lfloor = 1 \\ a_2 &= a_0 - qa_1 = 84 \\ y_2 &= y_0 + qy_1 = 1 \\ \\ i &= 2 \\ q &= \lfloor \tfrac{1}{a_2} \lfloor = 2 \\ a_3 &= a_1 - qa_2 = 5 \\ y_3 &= y_1 + qy_2 = 3 \\ \\ & i = 3 \end{aligned} \tag{19}$$

and then we stop since $a_3 \leq \lfloor \sqrt{\frac{p^r}{2}} \rfloor \equiv 5 \leq 11$. The answer is then given by

$$h = \frac{(-1)^{i-1} \cdot a_i}{y_i} = \frac{(-1)^{3-1} \cdot a_3}{y_3} = \frac{5}{3}. \tag{20}$$

We can now present Definitions 1 and 2.

Definition 1: The Hensel encode algorithm is denoted as

$$h = H(p, r, \frac{c}{d}) \tag{21}$$

for an odd prime number p, an arbitrary positive integer r and some rational number c/d such that

$$0 \leq |c| \leq N, \quad 0 \leq |d| \leq N, \quad N = \lfloor \sqrt{\frac{p^r}{2}} \rfloor. \tag{22}$$

Definition 2: The Hensel decode algorithm is denoted as

$$\tfrac{c}{d} = H^{-1}(p, r, h) \tag{23}$$

for an odd prime number p, an arbitrary positive integer r and a Hensel code h is given by:

let $a_0 = p^r$, $a_1 = m_a$, $y_0 = 0$ and $y_1 = 1$ and $i = 1$, then, while

doi: 10.18178/ijiee.2025.15.3.1

$$a_i \leq \lfloor \sqrt{\frac{p^r}{2}} \rfloor \tag{24}$$

is true, the following is computed:

$$\begin{aligned} q &= \lfloor \tfrac{a_0}{a_1} \rfloor \\ a_{i+1} &= a_{i-1} - q a_i \\ y_{i+1} &= y_{i-1} + q y_i \\ i &= i + 1 \end{aligned} \tag{25}$$

So the result is given by

$$\frac{c}{d} = \frac{(-1)^{i-1} \cdot a_i}{y_i}. \tag{26}$$

Theorem 3: For all odd prime numbers p, arbitrary positive integers r and rational numbers c/d such that

$$0 \leq |c| \leq N, \quad 0 < |d| \leq N, \quad N = \lfloor \sqrt{\frac{p^r}{2}} \rfloor, \tag{27}$$

given a Hensel code $h = H(p, r, c/d)$, it holds that

$$H^{-1}(p, r, H(p, r, \tfrac{c}{d})) = \frac{c}{d}. \tag{28}$$

Proof 4: Theorems 1 and 2 state that rational numbers encoded as in Definition 1 have a unique and correct Hensel code for which a Hensel decoding procedure as in Definition 2 will result in a unique and correct rational number. Observing the conditions in Theorems 1 and 2, the mapping between that particular set of rational numbers and their corresponding Hensel codes is isomorphic.

C. Secret Hensel Codes

We extend the notion of Hensel code to what we call *secret Hensel code* by keeping the prime number p private. In the scope of this work, we might implement secret Hensel codes in two different ways:

- Hensel codes as a member of $\mathbb{Z}_g$, where g is a product of primes;
- Hensel codes for computation without modulo reduction.

1) Secret hensel code I

Let c/d be a rational number, $p_1, p_2, \ldots, p_k$ be k odd prime numbers and r be an arbitrary positive integer. Let g be such that

$$g = \prod_{i=1}^{k} p_i^r. \tag{29}$$

We generate the secret Hensel code h in the conventional way, using only the first prime number of the k primes available:

$$h = cd^{-1} \bmod p_1^r \tag{30}$$

without ever disclosing p_1^r, $i = 1 \ldots k$. We let $h \in \mathbb{Z}_g$, where g is public. This means that we confine the computation with Hensel codes in $\mathbb{Z}_g$ as opposed to $\mathbb{Z}_{p_1^r}$. The conversion from Hensel code to a rational number requires the knowledge of p_1. We show next that without knowing p_1, one cannot uniquely and correctly recover c/d.

Theorem 4: A secret Hensel code calculated with a prime number p_1 that is made a member of $\mathbb{Z}_g$ where g is the product of k primes including p_1, cannot be uniquely solved by the EEA.

Proof 5: Theorem 2 states that $|c|, |d| \leq \lfloor \sqrt{\frac{p^r}{2}} \rfloor$ is required for finding unique solutions for c and d. The EEA requires knowing p^r to compute the convergents of h/p^r.

We now show that if one uses g as an argument in the EEA, replacing the private p_1^r, the computation will not correctly find c/d from h.

Theorem 5: Let a rational number c/d be also an integer, that is, $d = 1$, for all values of c such that $|c| \leq \lfloor \sqrt{\frac{p^r}{2}} \rfloor$, the Hensel code h of c/d is equal to c.

Proof 6: The proof for the theorem is implied in the Theorem 2.

Lemma 2: If $H(p_1, r, c/d) = h$ and g is a product of k prime numbers including p_1, and g is used as a replacement of p_1^r in the EEA in order to compute the rational number c/d from the corresponding Hensel code h, the following holds

$$H^{-1}(g, r, h) = h. \tag{31}$$

Proof 7: If $g = \prod_{i=1}^{k} p_i^r$ and a Hensel code $h = H(p_1, r, c/d)$ according to Definition 1, the following holds

$$h \leq \lfloor \sqrt{\frac{g}{2}} \rfloor \tag{32}$$

and according to Theorem 5, calculating the EEA for h and g will result in a rational number c/d where $c = h$ and $d = 1$, and therefore $H^{-1}(g, r, h) = h$.

Lemma 3: If c, d and p_1^r are unknown, there are infinitely many solutions for p_1^r given h and g, where $g = \prod_{i=1}^{k} p_i^r$ for $n \geq 2$.

Proof 8: The range of a Hensel code h under p_1^r is $0 \ldots p_1^r - 1$. For any $p_i^r \geq p_1^r$ there are infinitely many combinations of c, d and p^r that will result on the same h, since $\{0, \ldots, p_1^r - 1\} \subseteq \{0, \ldots, p_i^r - 1\}$.

The only efficient method known to date for the inverse map of Hensel codes is the slightly modified version of EEA (in the rational solution in the end of the computation), which we showed cannot be used for this task if p^r is unknown. One might be tempted to use gcd computations in an attempt to recover p^r, however, as remarked by Miola in [33], $h < p^r$ is a condition for a valid Hensel code with respect to p^r. By its nature, a prime number does not have any common divisor with numbers smaller than that prime number itself.

If one is able to find the prime factors of g, then a given Hensel code can be efficiently solved with the modified EEA. However, factoring large integers is known to be hard [50].

2) Secret hensel code II

Let c/d be a rational number, p be an odd prime number and r be an arbitrary positive integer. We generate the secret Hensel code h in the conventional way by calculating $m_a = cd^{-1} \bmod p^r$ without ever disclosing p^r where computations will be performed over Hensel codes similarly

doi: 10.18178/ijiee.2025.15.3.1

generated without modulo reduction. That means that everything that we stated about the Secret Hensel Code I remains true for Secret Hensel Code II with the exception of the prime factorization option from g, since that in this case, there is no g to be factored this time. In Section 4 we will demonstrate operations with and without the modulus g.

D. Multiple Hensel Codes

Calculating Hensel codes with multiple prime numbers is related to the notion of *g-adic numbers*. Its fundamental properties were discussed in details by Mahler in 1961 [51] and 1973 [52]. Morrison in [53] describes the g in g-adic as the product of k distinct primes such that

$$g = \prod_{i=1}^{k} p_i^r \tag{33}$$

where all g-adic numbers are considered expansions of rational numbers that are isomorphic to the set of rational numbers.

Theorem 2 states that there is an upperbound $N = \lfloor \sqrt{\frac{\overline{p^r}}{2}} \rfloor$ such that every rational number in which the absolute value of numerator and denominator, individually, is bounded by N, can be uniquely converted to its Hensel code and recovered back to rational form. The N in Theorem 2 is related to the fact that every rational number that has its numerator and denominator bounded by N in their absolute value is called an *order-N Farey fraction* [31], [32], [54].

In g-adic arithmetic, if g is a product of k odd prime numbers and r is a positive integer, then

$$N = \lfloor \sqrt{\frac{g}{2}} \rfloor, \; g = \prod_{i=1}^{k} p_i^r \tag{34}$$

and there is a one-to-one mapping from order- N Farey fractions into k digits Hensel codes. The concept of *digits* in Hensel code comes from the number of primes p in g and is directly related to the ability of encoding larger fractions where the greater the number of primes, the larger the order-N Farey fractions that can be uniquely encoded and decoded. In fact, one might chose which strategy to pursue: in order to encode larger fractions, one can increase the size of p, r, or use multiple p^r's (forming a g-adic number).

This concept is important in order to understand what we refer to as *multiple Hensel codes*. A Hensel code computed with a single prime number p is a *single digit Hensel code* while a Hensel code computed with k prime numbers $p_1, \ldots p_k$ is a *k-digit Hensel code*, thus, a g-adic number.

When a list of rational numbers is Hensel encoded, distinct instances of single digit Hensel codes are computed. We refer to the process of Hensel encoding a single rational number with multiple prime numbers as *multiple Hensel code*.

Definition 3: Given a rational number c/d , k prime numbers $p_1, \ldots p_k$, and an arbitrary positive integer r, the multiple Hensel encode

$$(h_1, \ldots, h_k) = H_g((p_1, \ldots, p_k), r, \frac{c}{d}) \tag{35}$$

is given by:

1. Calculate the individual Hensel codes of c/d with each pri me $p_1, \ldots, p_k$ in order to yield $(h_i, \ldots, h_k)$ such that

$$h_i = H(p_i, r, \frac{c}{d}), \; i = 1 \ldots k \tag{36}$$

2. Return $(h_i, \ldots, h_k)$.

Definition 4: Given multiple Hensel codes $(h_1, \ldots, h_k)$, k prime numbers $p_1, \ldots p_k$, and an arbitrary positive integer r, a multiple Hensel decode

$$\frac{c}{d} = H_g^{-1}((p_i, \ldots, p_k), r, (h_1, \ldots, h_k)) \tag{37}$$

is given by:

1. Let the total residue s be such that

$$s = \sum_{i=1}^{k} \frac{g}{p_i^r} \left(\frac{g}{p_i^r}\right)^{-1} \bmod p_i^r \, h_i \bmod g \tag{28}$$

where $g = \prod_{i=1}^{k} p_i^r$.

2. Apply the single digit Hensel decode to recover c/d

$$\frac{c}{d} = H^{-1}(g, r, s) \tag{39}$$

3. Return c/d.

Theorem 6: For all k odd prime numbers $p_1, \ldots, p_k$, arbitrary positive integers r and rational numbers c/d such that

$$0 \le |c| \le N, \quad 0 \le |d| \le N, \quad N = \sqrt{\frac{g}{2}} \tag{40}$$

where

$$g = \prod_{i=1}^{k} p_i^r, \tag{41}$$

given a k-digit Hensel code

$$(h_1, \ldots, h_k) = H_g((p_1, \ldots, p_k), r, \frac{c}{d}), \tag{42}$$

it holds that

$$H_g^{-1}((p_i, \ldots, p_k), r, H_g((p_1, \ldots, p_k), r, \frac{c}{d})) = \frac{c}{d}. \tag{43}$$

Proof 9: Theorem 3 states the correctness of single digit Hensel codes and its inverses. The direct map of multiple Hensel codes is computed as a list of single digit Hensel codes and therefore follows the same rules stated in Theorems 1 and 2. The first step of the multiple Hensel decode consists of calculating the total residue s, which is done via Chinese Remainder Theorem where each prime number p_i is associated to its corresponding Hensel code h_i, therefore the inverse map is equivalent to the decoding process of the single digit Hensel code.

Although we do not cover parallel computation [55]-[57] in this work, we remark that there are several instances of the use of g-adic numbers for parallel processing [53], [58]-[63] that take advantage of the properties we discuss in this section.

doi: 10.18178/ijiee.2025.15.3.1

III. Encoding Scheme

Definition 5: The data encoding scheme is a tuple of three polynomial-time algorithms

$$(\texttt{Setup}, \texttt{Encode}, \texttt{Decode}) \tag{44}$$

such that

1. The $\texttt{Setup}$ is a probabilistic polynomial-time algorithm that takes λ as input and outputs a key secret $k = (p_1, p_2, p_3, p_4, r_1, r_2)$, where $p_1, \dots, p_4$ are λ-bit odd prime numbers uniformly generated from the set of all λ-bit prime numbers P_λ, r_1 and r_2 are arbitrarily selected from the set $\{1, \dots, 4096\}$ observing the condition $r_2 \geq 6r_1$, and a public modulus $g = (\prod_{i=1}^{3} p_i^{r1}) p_4^{r2}$. We write $(k, g) \twoheadleftarrow \texttt{Setup}(\lambda)$.
2. The $\texttt{Encode}$ is a probabilistic polynomial-time algorithm that takes a secret key k and a datum $m \in \mathbb{Z}_{p_1^{r_1}}$ as input and outputs a code β. We write $\beta \twoheadleftarrow \texttt{Encode}(k, m)$.
3. The $\texttt{Decode}$ is a deterministic polynomial-time algorithm that takes a secret key k and a code β as input and outputs the datum m. We write $m = \texttt{Decode}(k, \beta)$.

Definition 6: The Setup algorithm $(k, g) \twoheadleftarrow \texttt{Setup}(\lambda)$ is given by:

1. Given the parameter λ, uniformly generate four prime numbers $p_1, \dots, p_4$, each one of bit length λ:

$$\begin{aligned} p_1, \dots, p_4 &\twoheadleftarrow \mathrm{P}_\lambda, \\ r_1, r_2 &\twoheadleftarrow \{1, \dots, 4096\},\ r_2 \geq 6r_1 \end{aligned} \tag{45}$$

2. where the secret key k is defined as $k = (p_1, p_2, p_3 p_4, r_1, r_2)$.
3. Compute the public modulus g such that

$$g = (\prod_{i=1}^{3} p_i^{r1}) p_4^{r2} \tag{46}$$

4. Return (k, g).

Definition 7: The Encode algorithm $\beta \twoheadleftarrow \texttt{Encode}(k, m)$ is given by:

5. Uniformly generate two random numbers s_1, s_2 such that $s_1 \in \mathbb{Z}_{p_2^{r_1}}$ and $s_2 \in \mathbb{Z}_{p_3^{r_1}}$.
6. Given a datum $m \in \mathbb{Z}_{p_1^{r_1}}$, apply the inverse of the multiple Hensel encode on (m, s_1, s_2) using the primes (p_1, p_2, p_3) and the arbitrary positive integer r_1 in order to calculate α:

$$\alpha \in \left(N = \sqrt{\frac{p_1^{r1} p_2^{r1} p_3^{r1}}{2}} \right) \tag{47}$$

7. Calculate $\beta \in \mathbb{Z}_{p_4^{r_2}}$ such that

$$\beta = H(p_4, r_2, \alpha) \tag{48}$$

8. Return β.

Definition 8: The Decode algorithm $m = \texttt{Decode}(k, \beta)$ is given by

1. Calculate α with the inverse Hensel code on β using p_4 and r_2 such that:

$$\alpha = H^{-1}(p_4, r_2, \beta) \tag{49}$$

2. Recover m by calculating the multiple Hensel code of α:

$$m = H(p_1, r_1, \alpha) \tag{50}$$

3. Return m.

Theorem 7: For all secret key k output by $\texttt{Setup}$ and all data $m \in \mathbb{Z}_{p_1^{r_1}}$, it holds that

$$\texttt{Decode}(k, \texttt{Encode}(k, m)) = m. \tag{51}$$

Proof 10: The proof of the theorem is implied in the proofs for Theorems 3 and 6.

Fig. 1 and Fig. 2 illustrate the encoding and decoding flows.

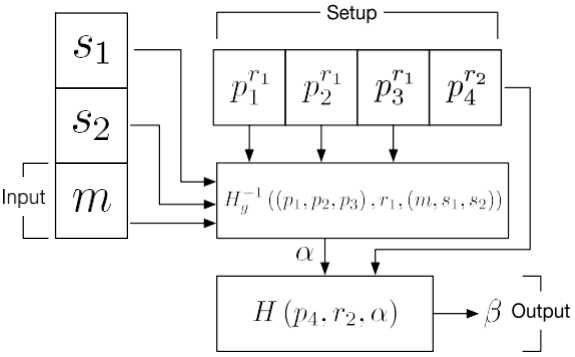

Fig. 1. Encoding.

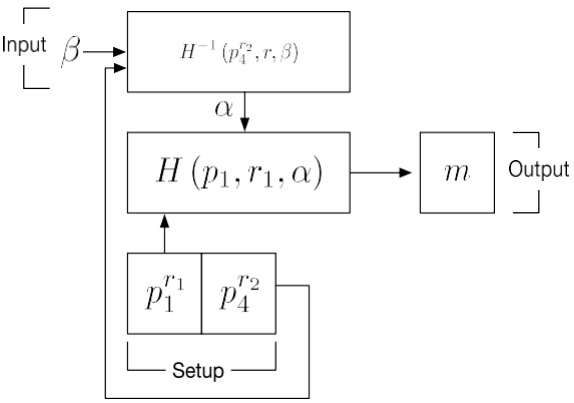

Fig. 2. Decoding.

IV. Examples

The encoding algorithm $\beta \twoheadleftarrow \texttt{Encode}(k, m)$ preserves addition and multiplication over encoded data, such that:

$$\begin{aligned} &\texttt{Encode}(k, m_1 + m_2) \\ &= \texttt{Encode}(k, m_1) + \texttt{Encode}(k, m_2) \end{aligned} \tag{52}$$

and

doi: 10.18178/ijiee.2025.15.3.1

$$\begin{aligned}&\texttt{Encode}(k, m_1 m_2)\\&= \texttt{Encode}(k, m_1)\texttt{Encode}(k, m_2)\end{aligned} \tag{53}$$

Subtraction and division are also possible however it requires adding a new encoding layer in our encoding scheme which enables any rational number as input (for subtraction) and the knowledge of p_4 (for division, since division is calculated via modular multiplicative inverse of the denominator and p_4). Thus, in order to keep encoding scheme with only two layers of encoding and keep p_1, p_2, p_3 private, one might develop desired operations with arithmetic circuits from combinations of addition and multiplication.

In this section we provide some examples of encoding, decoding and homomorphic operations with encoded data. For clarity, we will use the same setup configuration for all examples, which is:

$$((179,197,239,137,1,6), g) = \texttt{Setup}(8) \tag{54}$$

Next, we breakdown the encoding and decoding procedures.

Example 3: (Encoding data 1) Let $m_1 = 8$, $s_1 = 222$ and $s_2 = 191$. We first calculate α_1

$$\begin{aligned}\alpha_1 &= H_g^{-1}((p_1, p_2, p_3), r_1, (m, s_1, s_2))\\&= H_g^{-1}\left(\begin{pmatrix}179,\\197,\\239\end{pmatrix}, 1, \begin{pmatrix}23,\\222,\\191\end{pmatrix}\right)\\&= \frac{323}{3882}\end{aligned} \tag{55}$$

Then, β_1 is given by

$$\begin{aligned}\beta_1 &= H(p_4, r_2, \alpha_1)\\&= H\left(\frac{137,6,323}{3882}\right)\\&= 553542833964\end{aligned} \tag{56}$$

Example 4: (Encoding data 2) Let $m_2 = 19$, $s_1 = 132$ and $s_2 = 151$. We first calculate α_2

$$\begin{aligned}\alpha_2 &= H_g^{-1}((p_1, p_2, p_3), r_1, (m, s_1, s_2))\\&= H_g^{-1}\left(\begin{pmatrix}179,\\197,\\239\end{pmatrix}, 1, \begin{pmatrix}19,\\132,\\151\end{pmatrix}\right)\\&= \frac{184}{867}\end{aligned} \tag{57}$$

Then, β_2 is given by

$$\begin{aligned}\beta_2 &= H(p_4, r_2, \alpha_2)\\&= H\left(\frac{137,6,184}{867}\right)\\&= 6375446165524\end{aligned} \tag{58}$$

In the next sections we will demonstrate how to compute on β_1 and β_2 both without and with modulo reduction.

A. Computation Without Modulus

We will first compute addition and multiplication without reducing to modulus g. This can be useful for those who can afford an arbitrary size growth as a result of computations over encoded data. In this scenario, factoring g to find $p_1, \ldots, p_4$ is not an option, which would lead the attempts to solve for the secret prime numbers through techniques other than prime factorization.

Example 5: (Addition over encoded data 1 and 2)

$$\begin{aligned}&\beta_1 + \beta_2\\&= \begin{pmatrix}4858509755004 +\\4943268709105\end{pmatrix}\\&= 9801778464109\end{aligned} \tag{59}$$

we can see that

$$\begin{aligned}&\alpha_1 + \alpha_2 = H^{-1}(p_4, r_2, \beta_1 + \beta_2)\\&= H\left(\begin{vmatrix}137,\\6,\\9801778464109\end{vmatrix}\right)\\&= \frac{409749}{130663}\end{aligned} \tag{60}$$

and

$$\begin{aligned}m_1 + m_2 &= H(p_1, r_1, \alpha_1 + \alpha_2)\\&= H\left(\frac{179,1,409749}{130663}\right)\\&= 23\end{aligned} \tag{61}$$

therefore,

$$m_1 + m_2 = \texttt{Decode}(k, \beta_1 + \beta_2) = 8 + 15 = 23. \tag{62}$$

Example 6: (Multiplication over encoded data 1 and 2)

$$\begin{aligned}\beta_1\beta_2 &= \begin{pmatrix}4858509755004 \cdot\\4943268709105\end{pmatrix}\\&= 24016919244792672894111420\end{aligned} \tag{63}$$

and

doi: 10.18178/ijiee.2025.15.3.1

$$\alpha_1\alpha_2 = H^{-1}(p_4, r_2, \beta_1\beta_2)$$
$$= H^{-1}\left(\begin{matrix}137,\\ 6,\\ 240169192447926728941114201\end{matrix}\right)$$
$$= \frac{1246239}{522652} \tag{64}$$

and

$$m_1m_2 = H(p_1, r_1, \alpha_1\alpha_2)$$
$$= H\left(\frac{179,1,1246239}{522652}\right)$$
$$= 120 \tag{65}$$

therefore,

$$m_1m_2 = \mathrm{Decode}(k, \beta_1\beta_2) = 8 \cdot 15 = 120. \tag{66}$$

B. Computation with Modulus

Now we compute addition and multiplication modulo g. This is particularly relevant for applications that requires operations to be under a certain upper-bound. In this scenario, it is required that g is public.

(Addition over encoded data 1 and 2 modulo g)

$$\beta_1 + \beta_2 = \left(\begin{matrix}4858509755004 +\\ 4943268709105\end{matrix}\right) (\mathrm{mod}\ g)$$
$$= 9801778464109 \tag{67}$$

and

$$\alpha_1 + \alpha_2 = H^{-1}(p_4, r_2, \beta_1 + \beta_2)$$
$$= H^{-1}\left(\begin{matrix}137,\\ 6,\\ 9801778464109\end{matrix}\right)$$
$$= \frac{409749}{130663} \tag{68}$$

and

$$m_1 + m_2 = H(p_1, r_1, \alpha_1 + \alpha_2)$$
$$= H\left(\frac{179,1,409749}{130663}\right)$$
$$= 23 \tag{69}$$

(Multiplication over encoded data 1 and 2 modulo g)

$$\beta_1\beta_2 = \left(\begin{matrix}4858509755004 \cdot\\ 4943268709105\end{matrix}\right) (\mathrm{mod}\ g)$$
$$= 26226170778355423333 \tag{70}$$

and

$$\alpha_1\alpha_2 = H^{-1}(p_4, r_2, \beta_1\beta_2)$$
$$= H^{-1}\left(\begin{matrix}137,\\ 6,\\ 262261707783554233331\end{matrix}\right)$$
$$= \frac{1246239}{522652} \tag{71}$$

and

$$m_1m_2 = H(p_1, r_1, \alpha_1\alpha_2)$$
$$= H\left(\frac{179,1,1246239}{522652}\right)$$
$$= 120 \tag{72}$$

V. Features and Tread-Offs

A. Performance

The results in this section were obtained in an environment with the following specs:

- Processor: 2.8 GHz Intel Core i7
- Memory: 16 GB 1600 MHz DDR3
- OS: macOS High Sierra 10.13.6 (17G65)

The results in Tables I and II were obtained with the MSH Code Ruby library, ruby 2.6.3p62 (2019-04-16 revision 67580) [x86_64-darwin17]. Ruby is not an ideal programming language for writing software for performance tests. We chose Ruby for being a language of fast prototyping and human-friendly readability. The generation of prime numbers via OpenSSL implementation in Ruby (openssl 2.1.2) is very slow, thus the discrepancy in time from Setup algorithm to all the other operations. This should not be a critical factor since the Setup algorithm is the least executed algorithm among all the others (once the primes are generated, Setup will generally not be executed again for a while. The Setup algorithm consists mostly of just generating four prime numbers and computing their product. It is clear that the time taken to generate these primes is the determining factor for the overall algorithm runtime. To illustrate how the programming language might affect the runtime of Setup algorithm, consider the results in Table III for an implementation in Python 3.7.4.

TABLE I: Performance Results with $\lambda = 1024$

Algorithm/Operation	Time (seconds)
Setup	0.653246
Encode	0.006342
Decode	0.000038
Addition	0.000008
Multiplication	0.000019
Encoding 4D vector	0.012901
Decoding 4D vector	0.000176
Dot product 4D	0.000048

doi: 10.18178/ijiee.2025.15.3.1

TABLE II: PERFORMANCE RESULTS WITH $\lambda = 2048$

Algorithm/Operation	Time (seconds)
Setup	24.494334
Encode	0.009593
Decode	0.000055
Addition	0.000005
Multiplication	0.000027
Encoding 4D vector	0.034336
Decoding 4D vector	0.000179
Dot product 4D	0.000094

B. Compactness

Our construction consists of three small algorithms which will most likely result on small code bases no matter the programming language of choice. All the operations are performed over integers, including the computing of the upper-bound N (see Section 2.4) which is done via the integer implementation of the square root.

C. Encoding Size

The bit length of encoded data β using our construction is $|\beta|_{\text{bits}} \leq (r_2 \cdot |\lambda|_{\text{bits}})$. Operations over encoded data that are confined in $\mathbb{Z}_g$ will produce results where $\beta \leq g$. As an example, if $\lambda = 1024$, $r_1 = 1$ and $r_2 = 6$, encoding a datum m such that $|m|_{\text{bits}} = 8$ will generate a β such that $|\beta|_{\text{bits}} \leq 6144$. This expansion in size might be expensive for very hardware-limited environments, such as edge computing. However, we hope that the compactness and efficiency of our solution alongside the added benefits such as probabilistic and homomorphic encoding will enable opportunities that will justify the encoding size.

D. Secret Variable Management

The Setup algorithm yields four secret primes $p_1, \ldots, p_4$ which might be kept secret at all times. Applications that currently do not encode their data must consider the concerns of managing secret keys. Recommendations are found in .

TABLE III: SETUP ALGORITHM IN PYTHON 3

λ	Time (seconds)
1024	0.09601879119873047
2048	0.59281396865844730
4096	3.69716095924377440

VI. AVAILABILITY

Our implementation of the MSH Code Ruby library is available at https://github.com/davidwilliam/ruby-msh-code.

VII. CONCLUSIONS

In this work we introduced a probabilistic fully homomorphic data encoding scheme based on multiple secret Hensel codes. Our scheme can be used in conjunction with existing encryption tools or as an stand-alone hiding mechanism while offering promising obfuscation properties. These properties can be further analyzed and explored towards the realization of an encryption scheme. We discussed how we explored the foundations of p-adic and g-adic arithmetic in order to develop the concept of multiple secret Hensel codes that serves as the basis of a compact and efficient data encoding solution. While reviewing the properties of conventional Hensel codes, we discussed the opportunities of exploring secret parameters and the challenge of calculating the inverse map of a Hensel code without satisfying minimum requirements established in the literature. We showed how the notion of g-adic numbers can be useful to enable the combination of meaningless and meaningful data in the same partition, so we add a probabilistic feature to secret Hensel codes. We presented worked examples to demonstrate the homomorphic operations over encoded data and provided insights of how our construction can serve real-world applications, alongside a working code that implements our idea. Our solution can be extended to g-adic numbers of arbitrarily many k digits (the involved number of prime numbers). Not only we do not expect any significant loss in performance but further investigation might reveal an escalation in hardness by a factor of k. The same mathematical foundation used in our construction can be extended in order to provide other homomorphic solutions including parallel / distributed computation, multiparty computation and, by properly managing secret components within our scheme, a variety of security protocols.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

David W. H. A. da Silva conducted the research, proposed and implemented the constructions in this work and wrote the paper. Carlos Paz de Araujo reviewed the mathematical foundation and recommended the investigation and application of p -adic numbers for data computation and secrecy. Edward Chow supervised the content organization, the flow of ideas and the paper presentation; all authors had approved the final version.

REFERENCES

[1] J. K. Resch and W. B. Leggette, "Encoding data in a dispersed storage network," Google Patents, 2019.

[2] B. Conway and L. E. Halperin, "Data recovery utilizing optimized code table signaling," Google Patents, 2019.

[3] K. Narayanan, V. Honkote, D. Ghosh, and S. Baldev, "Energy efficient communication with lossless data encoding for swarm robot coordination," in *Proc. 2019 32nd International Conference on VLSI Design and 2019 18th International Conference on Embedded Systems (VLSID)*, 2019, pp. 525–526.

[4] R. M. Thanki and A. Kothari, "Data compression and its application in medical imaging," in *Hybrid and Advanced Compression Techniques for Medical Images*, Springer, 2019, pp. 1–15.

[5] J. C. Mogul, F. Douglis, A. Feldmann, and B. Krishnamurthy, "Potential benefits of delta encoding and data compression for HTTP," *ACM SIGCOMM Computer Communication Review*, vol. 27, no. 4, pp. 181–194, 1997.

[6] A. F. Naguib, N. Seshadri, and A. R. Calderbank, "Increasing data rate over wireless channels," *IEEE Signal Process. Mag.*, vol. 17, no. 3, pp. 76–92, 2000.

[7] G. K. Wallace, "The JPEG still picture compression standard," *IEEE Trans. Consum. Electron.*, vol. 38, no. 1, pp. xviii-xxxiv, 1992.

[8] D. L. Gall, "MPEG: A video compression standard for multimedia applications," *Commun. ACM*, vol. 34, no. 4, pp. 46–59, 1991.

[9] V. G. Cerf, "ASCII format for network interchange," 1969.

[10] F. Yergeau, "UTF-8, a transformation format of ISO 10646," 2003.

[11] Z. Chi, Y. Li, H. Sun, Y. Yao, and T. Zhu, "Concurrent cross-technology communication among heterogeneous IoT devices," *IEEE/ACM Trans. Netw.*, 2019.

doi: 10.18178/ijiee.2025.15.3.1

[12] R. M. Chao, H. C. Wu, C. C. Lee, and Y. P. Chu, "A novel image data hiding scheme with diamond encoding," *EURASIP J. Inf. Secur.*, vol. 2009, no. 1, p. 658047, 2009.
[13] E. T. Lin and E. J. Delp, "A review of data hiding in digital images," in *PICS*, 1999, vol. 299, pp. 274–278.
[14] M. T. Parvez and A. A.-A. Gutub, "Vibrant color image steganography using channel differences and secret data distribution," *Kuwait J Sci Eng*, vol. 38, no. 1B, pp. 127–142, 2011.
[15] T. Hwang and T. R. N. Rao, "Secret error-correcting codes (SECC)," in *Proc. on Advances in Cryptology*, 1990, pp. 540–563.
[16] P. Dutta, D. Bhattacharyya, and T. Kim, "Data hiding in audio signal: A review," *Int. J. Database Theory Appl.*, vol. 2, no. 2, pp. 1–8, 2009.
[17] M. R. Ogiela and U. Ogiela, "Grammar encoding in DNA-like secret sharing infrastructure," in *Advances in Computer Science and Information Technology*, Springer, 2010, pp. 175–182.
[18] G. G. Dagher, A. P. Machado, E. C. Davis, T. Green, J. Martin, and M. Ferguson, "Data storage in cellular DNA: Contextualizing diverse encoding schemes," *Evol. Intell.*, pp. 1–13, 2019.
[19] P. K. Naskar, S. Paul, D. Nandy, and A. Chaudhuri, "DNA encoding and channel shuffling for secured encryption of audio data," *Multimed. Tools Appl.*, pp. 1–24, 2019.
[20] L. R. Rabiner and B. Gold, "Theory and application of digital signal processing," *Englewood Cliffs*, NJ, Prentice-Hall, Inc., p.777, 1975.
[21] E. A. Lee, "Cyber physical systems: Design challenges," in *Proc. 2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, 2008, pp. 363–369.
[22] C. M. Bishop, *Pattern Recognition and Machine Learning*, springer, 2006.
[23] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, 2016.
[24] K. Hwang and A. Faye, "Computer architecture and parallel processing," 1984.
[25] B. Wickremasinghe, R. N. Calheiros, and R. Buyya, "Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications," in *Proc. 2010 24th IEEE International Conference on Advanced Information Networking and Applications*, 2010, pp. 446–452.
[26] K. Hensel, *Theorie der Algebraischen Zahlen*, vol. 1. BG Teubner, 1908.
[27] T. M. Rao and R. T. Gregory, "The conversion of Hensel codes to rational numbers," in *Proc. 1981 IEEE 5th Symposium on Computer Arithmetic (ARITH)*, 1981, pp. 10–20.
[28] T. M. Rao, "Conversion of Hensel codes to rational numbers," *Comput. Math. with Appl.*, vol. 10, no. 2, pp. 185–189, 1984.
[29] E. V. Krishnamurtht, "Matrix processors using p-adic arithmetic for exact linear computations," in *Proc. 1975 IEEE 3rd Symposium on Computer Arithmetic (ARITH)*, 1975, pp. 92–97.
[30] E. V. Krishnamurthy, T. M. Rao, and K. Subramanian, "Finite segmentp-adic number systems with applications to exact computation," in *Proc. the Indian Academy of Sciences-Section A*, 1975, vol. 81, no. 2, pp. 58–79.
[31] E. V. Krishnamurthy, "On the conversion of Hensel codes to Farey rationals," *IEEE Trans. Comput.*, vol. 100, no. 4, pp. 331–337, 1983.
[32] R. T. Gregory, "Error-free computation with rational numbers," *BIT Numer. Math.*, vol. 21, no. 2, pp. 194–202, 1981.
[33] A. Miola, "Algebraic approach to p-adic conversion of rational numbers," *Inf. Process. Lett.*, vol. 18, no. 3, pp. 167–171, 1984.
[34] X. Li, *Exact Matrix Computation by Multiple P-adic Arithmetic*, 2016.
[35] X. Li and C. Lu, "Proactive self-defense algorithm for large matrix calculation using multiple P-adic data type," in *Proc. the 2015 Conference on Research in Adaptive and Convergent Systems*, 2015, pp. 262–267.
[36] M. Etzel and W. Jenkins, "Redundant residue number systems for error detection and correction in digital filters," *IEEE Trans. Acoust.*, vol. 28, no. 5, pp. 538–545, 1980.
[37] Ç. K. Koç, "Parallel p-adic method for solving linear systems of equations," *Parallel Comput.*, vol. 23, no. 13, pp. 2067–2074, 1997.
[38] F. Winkler, "A p-adic approach to the computation of Gröbner bases," *J. Symb. Comput.*, vol. 6, no. 2–3, pp. 287–304, 1988.
[39] E. A. Arnold, "Modular algorithms for computing Gröbner bases," *J. Symb. Comput.*, vol. 35, no. 4, pp. 403–419, 2003.
[40] Y. LeCun, C. Cortes, and C. J. Burges, *MNIST Handwritten Digit Database*, 2010.
[41] B. S. Barillas, "Efficient machine learning inference for embedded systems with integer based restricted boltzmann machines classifiers," University of Colorado Colorado Springs. Kraemer Family Library, 2019.
[42] D. W. H. A. Da Silva, C. Paz De Araujo, and E. Chow, "Fully homomorphic key update and key exchange over exterior product spaces for cloud computing applications," in *Proc. IEEE Pacific Rim International Symposium on Dependable Computing, PRDC*, 2019, vol. 2019-Decem.
[43] D. W. H. A. da Silva, C. P. de Araujo, E. Chow, and B. Sosa Barillas, "A new approach towards fully homomorphic encryption over geometric algebra," in *Proc. 2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, 2019.
[44] D. W. H. A. da Silva, O. Hanes, E. Chow, B. Sosa Barillas, and C. P. de Araujo, "Homomorphic image processing over geometric product spaces and finite P-adic arithmetic," in *Proc. 2019 IEEE 11th International Conference on Cloud Computing Technology and Science (CloudCom)*, 2019.
[45] D. E. Knuth, "The art of computer programming, " *Pearson Education*, vol. 3, 1997.
[46] J. A. M. Naranjo, J. A. López-Ramos, and L. G. Casado, "Applications of the extended Euclidean algorithm to privacy and secure communications," in *Proc. 10th International Conference on Computational and Mathematical Methods in Science and Engineering*, 2010, pp. 702–713.
[47] J. E. H. A. V. Aho and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, 1974.
[48] D. E. Knuth, *The Art of Computer Programming*, Addison-Wesley, 1978.
[49] W. E. M. Hardy GH, *An Introduction to the Theory of Numbers*, Clarendon Press, 1979.
[50] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978.
[51] K. Mahler, "Lectures on diophantine approximations. Part I: g-adic numbers and Roths theorem," 1961.
[52] K. Mahler, *Introduction to P-Adic Numbers and Their Function*, CUP Archive, 1973.
[53] J. F. Morrison, "Parallel p-adic computation," *Inf. Process. Lett.*, vol. 28, no. 3, pp. 137–140, 1988.
[54] P. Kornerup and R. T. Gregory, "Mapping integers and Hensel codes onto farey fractions," *BIT Numer. Math.*, vol. 23, no. 1, pp. 9–20, 1983.
[55] L. G. Valiant, "A bridging model for parallel computation," *Commun. ACM*, vol. 33, no. 8, pp. 103–111, 1990.
[56] S. G. Akl, *Parallel Computation: Models and Methods*, vol. 4. Prentice Hall Upper Saddle River, 1997.
[57] D. Culler *et al.*, "LogP: Towards a realistic model of parallel computation," in *ACM Sigplan Notices*, 1993, vol. 28, no. 7, pp. 1–12.
[58] C. Limongelli and H. W. Loidl, "Rational number arithmetic by parallel p-adic algorithms," in *Proc. International Conference of the Austrian Center for Parallel Computation*, 1993, pp. 72–86.
[59] C. Limongelli, "On an efficient algorithm for big rational number computations by parallel p-adics," *J. Symb. Comput.*, vol. 15, no. 2, pp. 181–197, 1993.
[60] A. Colagrossi and C. Limongelli, "Big numbers p-adic arithmetic: A parallel approach," in *Proc. International Conference on Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes*, 1988, pp. 169–180.
[61] J. Tamura, "A class of transcendental numbers having explicit g-adic and Jacobi-Perron expansions of arbitrary dimension," *Acta Arith.*, vol. 71, no. 4, pp. 301–329, 1995.
[62] C. Limongelli, "Exact solution of computational problems via parallel truncated p-adic arithmetic," in *Advances in the Design of Symbolic Computation Systems*, Springer, 1997, pp. 68–83.
[63] J. Coates, "On p-adic L-functions attached to motives over ℚ II," *Bol. da Soc. Bras. Matemática-Bulletin/Brazilian Math. Soc.*, vol. 20, no. 1, pp. 101–112, 1989.

David W. H. A. da Silva received his B.S.B.A. degree in business administration from Universidade Potiguar in Natal, Rio Grande do Norte, Brazil in 2012, his M.S.C.S. in computer science from University of Colorado Colorado Springs, Colorado, USA, in 2017, and he is a PhD candidate in computer science also from University of Colorado Colorado Springs. His research is focused on special types of data transformations such as data representation, compression, and encryption.

doi: 10.18178/ijiee.2025.15.3.1

In 1998 he co-founded a Startup as a software engineer for developing e-commerces of the first generation in Brazil. In 2000 he co-developed one of the first sites of online gaming, receiving an award of Top 3 best site in Brazil in that year. In 2002 he became the main software engineer of Brazilian online News outlet and in 2004 he founded his second Startup, focused on R&D based on IoT and emerging technologies. In 2014 he joined ESPN Brazil and coordinated a team of software engineers responsible for all ESPN online and mobile products. In 2015 he decided to pause his career to invest in higher education. He is now on the last term of his PhD and work as a senior research scientist at Symetrix Corporation. Mr. Silva is an IEEE member and a member of the IEEE Computer Society.

Carlos Paz de Araujo received his BS/MS/PhD degrees in 1972-1982 in electrical engineering at the University of Notre Dame in South Bend, Indiana in the United States. He has been a professor of electrical engineering at the University of Colorado in Colorado Springs since receiving his PhD. He created Symetrix Corporation in 1989 and was a founder of Ramtron Corporation in 1984. Dr. Araujo became a IEEE fellow in 2012 "for Contributions to the Field of Ferroelectric Nonvolatile Memories". His work on Ferroelectric RAMs has been acknowledged with the 2006 IEEE Daniel Noble award. He has pioneered many new devices based on smart oxides. Some applications of these devices involved high dielectric constant nano-capacitors used in microwave devices and hearing aids. In the FeRAM development, he was the first to introduce fatigue free devices which led to many industrial partners and successful applications in smart cards like the JR Suica (Japan) and others. Over 1.3 Billion devices have been produced under Dr. Araujo's patents worldwide. He is also involved in ferroelectric-based advanced IR devices for medical applications. His current interest is in correlated electron devices as memory (created CeRAM, correlated electron Non-volatile RAM) and ultrafast switches for beyond CMOS technology. He is the author of over 300 papers and more than 200 US issued Patents. Also, he has over 300 foreign patents.

Edward Chow received his B.S. degree in 1977 from the National Taiwan University, and his M.A. and Ph.D. degrees from the University of Texas, Austin, in 1982 and 1985. He joined the Computer Science Department in 1991 as an associate professor, and became an Interim El Pomar Chair in October of 2002. He was appointed as a professor in July of 2004. Dr. Chow has taught a variety of courses, including Principles of computer science, web programming, software security, computer communications, multimedia computing and communications, advanced web systems and internet, fundamentals of computer/network security, distributed networks, and advanced system security designs.

He was the principal investigator of Secure Collective Network Defense Project, TNUA-UCCS International Cooperation on Human Body Movement Analysis project, Making Secure Information Sharing Easy and A Reality project, Network Restoration and Survivable Architecture project, as well as several other projects. He is presently working on web systems/security, WAN load balancing, internet network measurement, wireless network planning, high speed networks/protocols, distributed/optimal algorithms for network resource allocation, and interactive programming environments for network/protocol design.

doi: 10.18178/ijiee.2025.15.3.1