HERFA: A Homomorphic Encryption-based Root-Finding Algorithm

Christopher Bencini, Jason Mendola, Wei He, and Sunwoong Kim

***Abstract*—Edge-cloud computing architectures are exposed to significant security challenges. Although general encryption methods can mitigate some of these concerns, they require decryption to perform operations on data, exposing the data and secret keys to potential attacks. Homomorphic encryption (HE), which allows operations on encrypted data without decryption, provides an effective solution to this issue. HE-based root-finding algorithms can expand the use of HE to a wider range of real-world applications that involve solving equations. This letter presents an adaptation of the well-known Newton's method for use in the HE domain. Specifically, it employs a division-free approach to remove the division operation, which is not a basic HE operation. In addition, the proposed method is extended to handle a polynomial multiplicity greater than one for faster convergence. Compared to an alternative implementation that uses a numerical method for division, the proposed HE-based root-finding algorithm (HERFA) significantly reduces the number of sequential multiplications, which is a key factor limiting the feasibility of applications in the HE domain. This reduction allows HERFA to achieve faster execution speeds or higher accuracy.**

***Index Terms*—Homomorphic encryption, numerical method, root-finding algorithm, security.**

I. Introduction

THE global edge computing market is experiencing rapid growth, with an expanding array of real-world applications. However, this decentralized architecture presents significant security challenges. For example, data transmitted between edge devices and the cloud can be vulnerable to interception by malicious attackers. One effective solution is to apply encryption to data, and the advanced encryption standard (AES) has been successful in addressing these concerns. However, AES requires that encrypted data is decrypted before any operations are performed on it. If this decryption occurs on a third-party device, exposing the data and a secret key for decryption can lead to significant security risks.

Homomorphic encryption (HE) is a technique that enables computations on ciphertexts directly without decrypting them first [1]. There is no need to store a secret key and perform decryption on a third-party server, so HE has been widely adopted in many privacy-preserving real-world applications. These applications include machine learning [2], healthcare data analysis [3], and image/speech processing [4], [5].

Root-finding algorithms are widely used across diverse fields, such as control systems, computer graphics, and economic analysis, but their use in the HE domain remains largely unexplored. When implementing root-finding algorithms in the HE domain, two major limitations arise: 1) the types of operations permitted (e.g., only addition and multiplication) and 2) the number of sequential operations, particularly multiplications. Since root-finding algorithms generally involve non-polynomial operations, such as division and comparison operation, and iterative processes, adapting them to the HE domain presents several challenges.

In this letter, the popular Newton's method that finds one root of a real-valued function is moved to the HE domain. It avoids expensive comparison operations required in certain root-finding algorithms but does involve division, which is not a basic operation in HE schemes. To overcome this challenge, this letter adopts a division-free approach. This method is refined to enhance convergence when a polynomial multiplicity is known. In addition, the multiplication order is optimized to minimize overflow, and multithreading is applied to improve execution time. The proposed HE-based root-finding algorithm (HERFA) is analyzed against an alternative HE-based implementation of Newton's method that uses a numerical method for division. To the best of the authors' knowledge, this study is the first on a root-finding algorithm for homomorphically encrypted data that avoids non-polynomial operations.

II. Background

A. Homomorphic Encryption

HE schemes are categorized into bit-wise and word-wise schemes [6]. Generally, bit-wise HE schemes provide logical operations of messages in the HE domain, while word-wise HE schemes support additions and multiplications of messages. A word-wise HE scheme, the focus of this letter, involves the following algorithms:

- `KeyGen(params)`: takes HE parameters and generates a secret key `sk`, a public key `pk`, and an evaluation key `evk`.
- `Enc`(μ, `pk`): generates a ciphertext `ct` from a plaintext message μ using `pk`.
- `Dec(ct, sk)`: generates a plaintext message μ from a ciphertext `ct` using `sk`.
- `HomAdd`($\texttt{ct}_1$, $\texttt{ct}_2$): adds ciphertexts $\texttt{ct}_1$ and $\texttt{ct}_2$ of plaintext messages μ_1 and μ_2 and generates a ciphertext of a message $\mu_1 + \mu_2$.
- `HomMul`($\texttt{ct}_1$, $\texttt{ct}_2$, `evk`): multiplies ciphertexts $\texttt{ct}_1$ and $\texttt{ct}_2$ of plaintext messages μ_1 and μ_2 using an `evk` and generates a ciphertext of a message $\mu_1 \cdot \mu_2$.

Note that `HomSub`, which is a variant of `HomAdd`, is also commonly supported in word-wise HE schemes.

Manuscript received... *(Corresponding author: Sunwoong Kim.)*

C. Bencini, J. Mendola, W. He, and S. Kim are with the Department of Electrical and Microelectronic Engineering, Rochester Institute of Technology, Rochester, NY 14623 USA (email: {cmb3030, jpm4675, wh9297, sskeme}@rit.edu).

Algorithm 1 $\texttt{Inv}(x; d)$ [6]

Input: $0 < x < 2$, $d \in \mathbb{N}$
Output: an approximate value of $1/x$

$a_0 \leftarrow 2 - x$
$b_0 \leftarrow 1 - x$
for $(i = 0;\ i < d;\ i = i + 1)$ **do**
  $b_{i+1} \leftarrow b_i^2$
  $a_{i+1} \leftarrow a_i \cdot (1 + b_{i+1})$
end for
return a_d

HE schemes have a constraint known as the (multiplicative) depth, which refers to the number of sequential HomMuls. For instance, $x_1 \cdot x_2 + x_3 \cdot x_4$ requires a depth of 1, while $(x_1 \cdot x_2 + x_3) \cdot x_4$ requires a depth of 2. The maximum depth represents the highest number of sequential HomMuls allowed, which ensures that the resulting ciphertext is still correctly decrypted. This maximum depth is determined based on HE parameters. All HomMuls must be completed before reaching the maximum depth. Note that this letter does not consider the use of bootstrapping, which allows for an unlimited number of HomMuls through recryption, due to its extremely high computational complexity.

To enable non-polynomial operations, such as division and comparison operation, in the HE domain, numerical methods have been adopted [4], [6], [7]. For example, Cheon *et al.* introduced a numerical method that computes the inverse or reciprocal of x using d iterations, tailored specifically for the Cheon-Kim-Kim-Song (CKKS) scheme [8], as detailed in Algorithm 1 [6]. Each iteration involves performing two HomMuls and one HomAdd in the HE domain. It increases the total depth by 1 because a_i and $1 + b_{i+1}$, both having a depth of $i+1$ (when $i > 0$), are multiplied together to compute a_{i+1}, which increases the total depth to $i + 2$.

B. Newton's Method

Newton's method, also known as the Newton-Raphson method, is one of the most widely used algorithms that find approximations to the roots of real-valued functions [9]. It is simple to implement and only requires the function along with its derivative. It begins with an initial guess for the root, denoted by x_0, and iteratively refines this estimate. When a good initial guess is provided, this method shows quadratic convergence near the root. The formula for this method, given an input function $f(x)$, is as follows:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \tag{1}$$

where x_n stands for the approximation to the root of f after n iterations, and $f'(x_n)$ must be non-zero.

If the root of a polynomial function has a multiplicity m greater than one (i.e., $f(x) = (x - r)^m \cdot g(x)$, where $m > 1$ and $g(r) \neq 0$), the convergence rate of Newton's method drops to linear. However, if m is known, the formula of Newton's method can be modified as (2) to improve efficiency [9].

$$x_{n+1} = x_n - m \cdot \frac{f(x_n)}{f'(x_n)}. \tag{2}$$

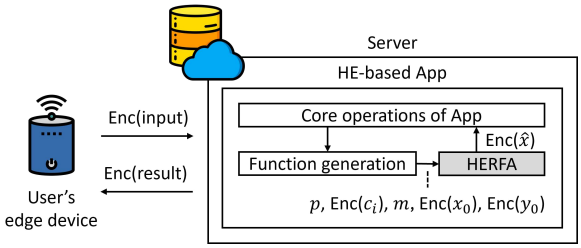

Fig. 1. A scenario where HERFA is used as part of a HE-based application.

This modification restores the quadratic convergence rate.

III. Newton's Method for Encrypted Data

Applying HE schemes to Newton's method introduces several challenges. These challenges include calculating derivatives and performing division in the HE domain. In this letter, it is assumed that the input function for Newton's method is a polynomial, which simplifies derivative calculations.

To address the issue related to the division in Newton's method, this letter adopts the two-step, division-free approach proposed by Blanchard and Chamberland [10]. The formulas for this method are as follows:

$$y_{n+1} = y_n \cdot (2 - f'(x_n) \cdot y_n), \tag{3}$$
$$x_{n+1} = x_n - y_{n+1} \cdot f(x_n). \tag{4}$$

Equation (3) provides an approximation for $1/f'(x_n)$ in (1), and y_0 is defined as $1/f'(x_0)$. For instance, the root of $f(x) = x^3 - 4x^2 + 1$ within the interval [0, 2] is 0.537. Using the standard Newton's method with an initial guess x_0 of 2, the iterative results are $x_1 = 0.25$, $x_2 = 0.672$, $x_3 = 0.547$, and $x_4 = 0.537$. In contrast, the division-free method with the same initial guess produces the following pairs: (y_1, x_1) = (−0.25, 0.25), (y_2, x_2) = (−0.387, 0.546), (y_3, x_3) = (−0.254, 0.538), and (y_4, x_4) = (−0.286, 0.537), which shows that the division-free method finds a comparable root.

To support a polynomial multiplicity greater than 1, this letter modifies (3) into (5) by combining (2) and the steps described in [10].

$$y_{n+1} = y_n \cdot (1 + m - f'(x_n) \cdot y_n). \tag{5}$$

It does not increase the total depth but enables quadratic convergence when the multiplicity of a root is known.

HERFA is implemented by translating the operations in (4) and (5) into the basic HE operations introduced in Section II-A. Fig. 1 shows its possible use case, where "Enc" stands for encryption. In this scenario, a user's edge device encrypts data, which is an input for a real-world application on a cloud server, using a public key and transmits it. Core homomorphic operations of the application process this encrypted data without decryption, and a polynomial function is generated for a specific purpose, such as modeling and optimization, in the application. A generated function is represented by a polynomial degree p, encrypted coefficients c_i $(0 \leq i \leq p)$, and a polynomial multiplicity m. These components, along with an encrypted initial guess x_0 and an encrypted y_0 for (5), are sent to HERFA. The calculated root $\hat{x}$ through n iterations, which is still encrypted, is then used in the application to

TABLE I
TOTAL DEPTH OF HE-ALTERNATIVE

Stage	Additional Depth
$f'(x_n)$ calculation	$\lceil \log_2 p \rceil$
$s \cdot (f'(x_n))^2$ calculation	2
`Inv` calculation	$d+1$
Remainder	1
Total (n iterations)	$n \cdot (\lceil \log_2 p \rceil + d + 4)$

TABLE II
TOTAL DEPTH OF HERFA

Stage	Additional Depth
$f'(x_n)$ calculation	$\lceil \log_2 p \rceil$
y_{n+1} calculation	1
x_{n+1} calculation	1
Total (n iterations)	$n \cdot (\lceil \log_2 p \rceil + 2)$

fulfill its intended purpose. The final result of the application is sent back to the user, who decrypts it using a secret key. Throughout the entire process, the cloud server remains unable to obtain any information about the user's input.

IV. DEPTH ANALYSIS

This section analyzes the total depth of HERFA. To demonstrate the effectiveness of HERFA, an alternative HE-based implementation of Newton's method using a numerical method for division is first introduced. It is denoted by HE-Alternative in the rest of this letter. In HE-Alternative, $f(x_n)/f'(x_n)$ in (2) is approximated to $f(x_n) \cdot \texttt{Inv}(f'(x_n); d)$. However, this approximation has challenges due to the input constraints of Algorithm 1. Specifically, $f'(x_n)$ must be 1) a positive real number and 2) less than 2. To address these challenges, (2) is modified as (6) in HE-Alternative.

$$x_{n+1} = x_n - m \cdot s \cdot \frac{f(x_n) \cdot f'(x_n)}{s \cdot (f'(x_n))^2}, \tag{6}$$

where s is a constant in the interval $(0, 1)$. The denominator $s \cdot (f'(x_n))^2$ is used as an input of `Inv`.

Table I shows the total depth of HE-Alternative, where `Inv` uses d iterations and Newton's method itself uses n iterations (a total of $d \times n$ iterations). In calculating $f'(x_n)$, $(c_p \cdot p) \cdot x^{p-1}$ is computed using a balanced binary tree approach, adding a depth of $\lceil \log_2 p \rceil$. The subsequent stage, which computes $s \cdot f'(x_n) \cdot f'(x_n)$, contributes an additional depth of 2. Each iteration of `Inv` increases the total depth by 1, except for the first iteration that increases the total depth by 2. Therefore, `Inv` adds an additional depth of $d+1$. In the rest of each iteration of Newton's method, the multiplication between $m \cdot s \cdot f(x_n) \cdot f'(x_n)$ and the `Inv` result increases the total depth by 1. The total depth increases proportionally to n.

The total depth of HERFA is shown in Table II. Unlike HE-Alternative, which requires the computation of $s \cdot (f'(x_n))^2$ and `Inv`, HERFA calculates y_{n+1} as described in Section III. It requires the depth of $\lceil \log_2 p \rceil + 1$ due to a multiplication between y_n^2 and $f'(x_n)$. The calculation of x_{n+1} adds another depth of 1 due to $y_{n+1} \cdot f(x_n)$. After n iterations, the total depth is $n \cdot (\lceil \log_2 p \rceil + 2)$, which is reduced by $n \cdot (d+2)$ compared to HE-Alternative.

TABLE III
TEST PROBLEMS USING HERMITE POLYNOMIALS

No	Polynomial	Root (near x_0)
P1	$x^2 - 1$	1.00000
P2	$x^3 - 3x$	1.73205
P3	$x^4 - 6x^2 + 3$	2.33441
P4	$x^5 - 10x^3 + 15x$	2.85697
P5	$x^6 - 15x^4 + 45x^2 - 15$	3.32426
P6	$x^7 - 21x^5 + 105x^3 - 105x$	3.75044

V. OPTIMIZATIONS

This letter presents two optimizations for HERFA. First, we optimize the multiplication order to minimize overflow when working with high-degree polynomials, based on the fact that y_{n+1} approximates $1/f'(x_n)$. For example, when y_n is multiplied by $f'(x_n) = 4 \cdot x_n^3$ for the computation in (5), we choose the order $((4 \cdot y_n) \cdot (x_n \cdot x_n)) \cdot x_n$ instead of $y_n \cdot ((4 \cdot x_n) \cdot (x_n \cdot x_n))$, which does not increase the total depth. Second, we apply multithreading to HERFA. The most time-consuming component of HERFA is the computation of $f'(x_n)$ and $f(x_n)$ in the HE domain, which are independent of each other. Therefore, different threads are used for these computations. In addition, we exploit parallelism between calculations of terms of $f'(x_n)$ and $f(x_n)$ and within the calculation of each individual term.

VI. EVALUATION

This section evaluates HERFA in terms of accuracy, total depth, and execution time. To implement HERFA, Microsoft SEAL version 3.6 is used [11]. Specifically, the functions for the CKKS scheme [8] are used to support real number roots of a function. The HE parameters used are as follows:

- Security level: 128 bits
- Polynomial degree: 2^{15}

Bits for primes are configured to support a maximum depth of 20, with the first and last primes using relatively more bits.

A. Accuracy

To evaluate the accuracy, six probabilist's Hermite polynomials [12], which are widely used in probability theory and systems theory, are employed as test polynomials. They are shown in Table III. The value of x_0 is set to 5, which is greater than the roots of all the test polynomials. The roots nearest to x_0, which are the targets, are listed in the last column.

To the best of the authors' knowledge, no previous work has been presented on root-finding algorithms for homomorphically encrypted data. Therefore, the standard non-HE-based Newton's method and HE-Alternative, which combines the standard Newton's method with Algorithm 1 in the HE domain, are used for comparison. The number of iterations n for Newton's method changes from 2 to 4, and the number of iterations d for `Inv` of HE-Alternative is fixed at 2.

Table IV shows the relative errors to the target roots (%). As the test polynomials progress from P1 to P6, the relative errors tend to decrease because the target roots are nearer to x_0. Compared to the standard non-HE-based Newton's method, HERFA results in larger errors. These errors are caused by

TABLE IV
RELATIVE ERRORS (%)

No	Non-HE-based			HE-based					
	Standard Newton			HE-Alternative			HERFA		
	n=2	n=3	n=4	n=2	n=3	n=4	n=2	n=3	n=4
P1	49.2	8.1	0.3	58.1	31.1	19.2	74.8	**29.7**	**8.3**
P2	45.7	15.2	2.6	53.2	38.9	31.2	61.7	**35.5**	**18.0**
P3	35.3	14.6	3.7	41.5	33.1	28.3	45.5	**29.4**	**17.3**
P4	25.5	11.2	3.2	28.7	23.1	19.9	32.4	**21.7**	**13.5**
P5	17.3	7.5	2.0	20.7	17.0	14.9	22.0	**14.7**	**9.1**
P6	10.6	4.1	0.9	11.7	9.0	7.5	13.8	**8.9**	**5.1**

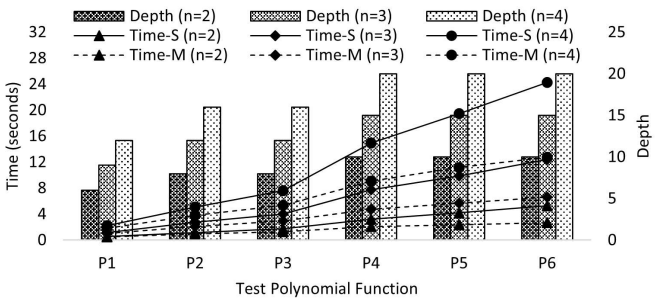

Fig. 2. Total depth and execution time (seconds) of HERFA depending on the number of iterations of Newton's method.

the use of the division-free approach. The transition to the HE domain introduces negligible errors.

Compared to HE-Alternative, HERFA has three key advantages. First, in this experiment, the value of s for HE-Alternative is empirically chosen to meet the input constraints of $\texttt{Inv}(s \cdot (f'(x_n))^2; d)$. However, in practice, this is not feasible because $(f'(x_n))^2$ is encrypted and unknown, so the s value must be set sufficiently small. This underlines the advantage of HERFA, which does not require s.

Second, as n increases, the efficiency of HE-Alternative decreases due to errors from $\texttt{Inv}$. Specifically, HE-Alternative, which is basically based on the standard Newton's method, produces lower errors than HERFA when n is 2. However, as n exceeds 2, the input value of $\texttt{Inv}$ becomes almost 0 due to the multiplication with s, leading to an increase in approximation errors of $\texttt{Inv}$. Consequently, HERFA yields smaller errors than HE-Alternative, as emphasized by the bold fonts.

Third, HE-Alternative becomes infeasible for n values of 3 and 4 under the HE parameter set used in this experiment because the total depth exceeds the maximum allowed. For instance, when n is 3 and p is 2, the total depth reaches 21 $(= 3 \cdot (\lceil \log_2 2 \rceil + 2 + 4))$. In contrast, HERFA supports these n values. For example, when n is 4 and p is 7, the total depth is 20 $(= 4 \cdot (\lceil \log_2 7 \rceil + 2))$, which meets the depth constraint.

B. Total Depth and Execution Time

The implemented HERFA software runs on a workstation with an Intel Xeon W-2255 processor and 64GB of RAM. We configured this software to use up to 8 threads. Fig. 2 shows the total depth and execution time results of HERFA based on the number of iterations in Newton's method. Time-S and Time-M represent execution times for single-threaded and multi-threaded operations, respectively. Note that HE-Alternative, which is used for accuracy comparison, is not included in this experiment because it is infeasible for $n = 3$ and $n = 4$ under the HE parameters.

As the test polynomials progress from P1 to P6, the execution time increases. In particular, a slightly steeper slope is observed when the total depth changes, particularly between P3 and P4. As n increases, the operations in (4) and (5) are performed more, leading to longer execution times. The application of multithreading reduces the average execution time by 32%. When the number of threads is limited to 2, the improvement becomes 27%. Notably, P6, which has the highest levels of parallelism, achieves an improvement of over 48% with more than two threads and 38% with two threads.

VII. CONCLUSION

In this letter, we adapt Newton's method to the HE domain to find the roots of encrypted polynomial functions. The division-free approach used is well-suited for the HE domain as it removes division. The depth analysis and experimental results demonstrate that the proposed method significantly reduces total depth–and consequently execution time–while reducing errors of an alternative implementation where a numerical method for division generates additional errors. For future work, we plan to adopt a hybrid approach, which uses different types of root-finding algorithms together, to mitigate sensitivity to the initial guess of Newton's method.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grant No. 2347253. We are grateful to the RIT FEAD and SURF grant programs.

REFERENCES

[1] J. H. Cheon *et al.*, "Introduction to Homomorphic Encryption and Schemes," in *Protecting Privacy through Homomorphic Encryption*, Springer, 2021, pp. 3–28.
[2] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy," in *Proc. ICML*, 2016.
[3] L. Jiang, L. Chen, T. Giannetsos, B. Luo, K. Liang, and J. Han, "Toward Practical Privacy-Preserving Processing Over Encrypted Data in IoT: An Assistive Healthcare Use Case," *IEEE Internet Things J.*, vol. 6, no. 6, Dec. 2019.
[4] S. Kim and W. Cho, "Accelerating Homomorphic Comparison Operations for Thresholding Using an Asymmetric Input Range and Input Scaling," in *Proc. GLSVLSI*, 2024.
[5] D. L. Elworth and S. Kim, "HEKWS: Privacy-Preserving Convolutional Neural Network-based Keyword Spotting with a Ciphertext Packing Technique," in *Proc. MMSP*, 2022.
[6] J. H. Cheon, D. Kim, D. Kim, H. H. Lee, and K. Lee, "Numerical Method for Comparison on Homomorphically Encrypted Numbers," in *Proc. ASIACRYPT*, 2019.
[7] J. H. Cheon, D. Kim, and D. Kim, "Efficient Homomorphic Comparison Methods with Optimal Complexity," in *Proc. ASIACRYPT*, 2020.
[8] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic Encryption for Arithmetic of Approximate Numbers," in *Proc. ASIACRYPT*, 2017.
[9] G. Dahlquist and Å. Björck, *Numerical methods*, Chelmsford, MA, USA: Courier Corporation, 2003.
[10] J. D. Blanchard and M. Chamberland, "Newton's Method Without Division," *The American Mathematical Monthly*, vol. 130, no. 7, pp. 606-617, Apr. 2023.
[11] Microsoft SEAL (release 3.6), [Online]. Available: https://github.com/Microsoft/SEAL
[12] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, 10th ed. New York, NY, USA: Academic, 1972.