Extension of the CKKS Cryptosystem: Part 1

Introduction to Homomorphic Cryptosystems - Lecture 6

Addition and Multiplication are not enough!

- CKKS (and other FHE encryption schemes) only allow addition and multiplication on the ciphertexts
- In the real world we need more then that!

Example: Division

What is the problem here?

```
#integer division a/b
def div(a,b):
  result = 0
  while(a > 0):
    a = a - b
    result++
  return result

div(15,5) -> 3
```

Answer:
We cannot calculate this homomorphically!
How can we know, that a number is bigger than another number?

We need other methods to implement more advanced functions like division, square root etc.

Mathematical Functions of Interest

DIVISION

SQUARE ROOT

EXPONENTIAL FUNCTION & LOGARITHM

MAXIMUM AND MINIMUM

DIVISION

Methods

We already know, that we cannot compute the division in the conventional way (iterative subtraction).

Based on our limitations in CKKS, there are only two suitable methods in literature:

We favour Newton-Raphson as it requires a smaller multiplicative depth.

Goldschmidt
Iterative multiplication of dividend and divisor with a common factor f_i:

$$q = \frac{a}{b} * \frac{f_1}{f_1} * \frac{f_2}{f_2} * \cdots$$

The goal is to make b converge to 1, so

$$f_{i+1} = 2 - b_i \quad (0 < b < 1)$$

When b_i is close to 1 you geht the result of the division through $a_i = q$

Newton-Raphson
Find the reciprocal of b with Newton's method so that we can calculate:

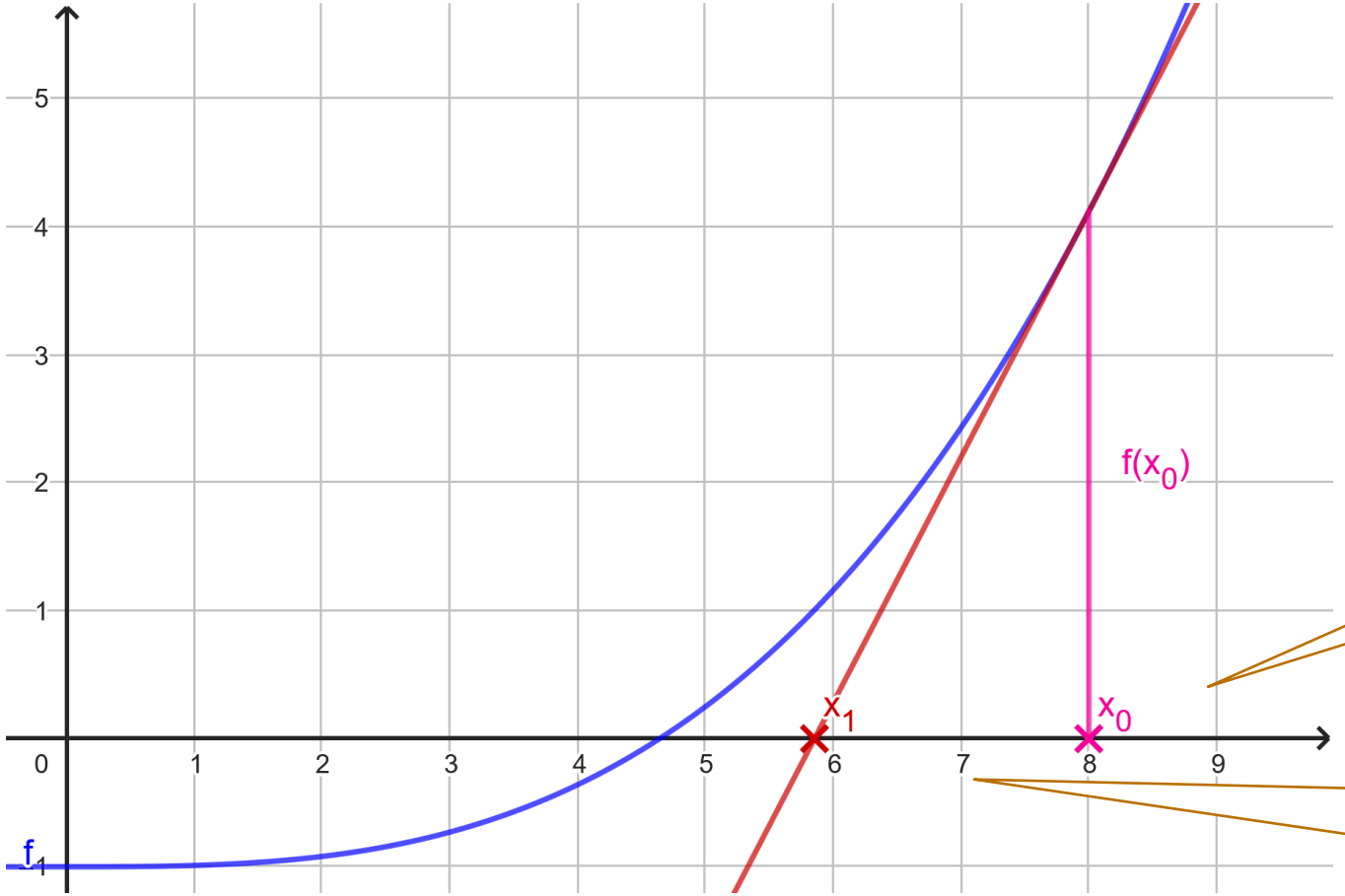
$$q = \frac{a}{b} = a * \frac{1}{b}$$

Newton's method can give us the x-value (root) for which a function $g(x)$ is zero by iteratively computing

$$x_{n+1} = x_n - \frac{g(x)}{g'(x)}$$

Intuition behind Newton-Raphson method

How does the Newton-Raphson method work?

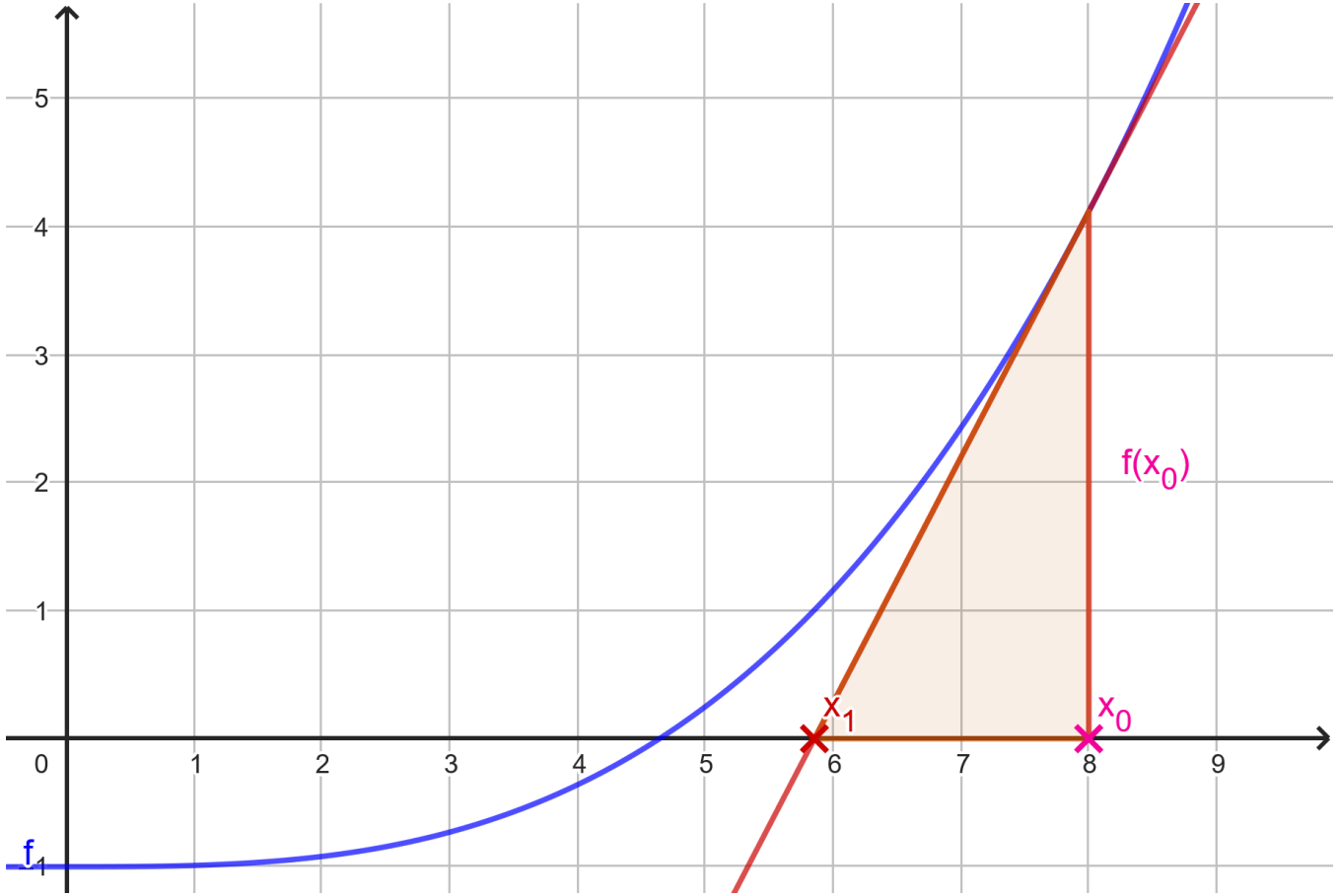


We need to choose a start approximation for the x-value.

x_1 is a better approximation then x_0, because the tangent line approximates the function

Intuition behind Newton-Raphson method

How does the Newton-Raphson method work?



How to calculate x_1?

Imagine the “slope triangle” of the tangent line.
It can be expressed as

$$f'(x_0) = \frac{f(x_0) - 0}{x_0 - x_1}$$

Solving for x_1:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

After multiple iterations, we reach the desired root.

Back to Divison

Reminder

We want to find the reciprocal of a value b in to calculate $a * \frac{1}{b} = \frac{a}{b}$

Idea

Define a function $g(x)$ which has its root at $\frac{1}{b}$ and use Newton's method.

Simple choice for $g(x)$:

$$g(x) = \frac{1}{x} - b$$

This satisfies our requirements as $g\left(\frac{1}{b}\right) = 0$.

This is very convenient!
We can calculate the next x-value using only addition and multiplication.
So this can be done on encrypted values using CKKS.

Newton's method

With $g(x)$, Newton's method gives us this for x_{n+1}:

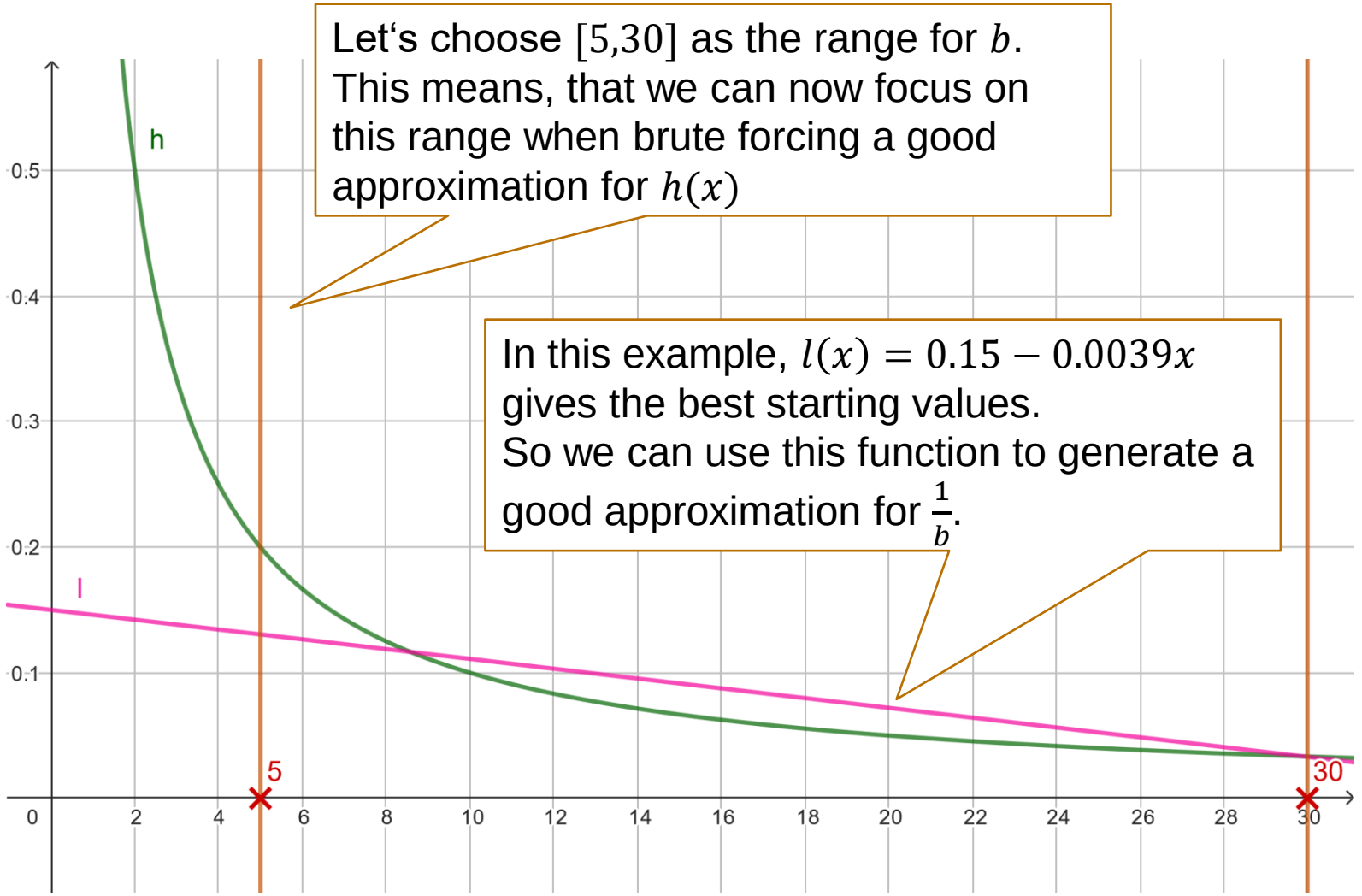
$$x_{n+1} = x_n - \frac{g(x_n)}{g'(x_n)} = x_n - \frac{\frac{1}{x_n} - b}{\frac{-1}{x_n^2}} = x_n + x_n - x_n^2 b = x_n(2 - x_n b)$$

But how to choose x_0?

Choosing a starting value

Problem
We need to find an approximation for $\frac{1}{b}$ as a starting value for the Newton method.
But: We only have the encrypted value of b.

Idea
Given the range $[o, u]$ of b we can use brute force to find a good enough linear auxiliary function $l(x)$ to calculate $h(x) = \frac{1}{x}$ using Newton.



Division – Bringing it all together

To calculate the division homomorphically, we need these steps:

The parameters for linear approximation are calculated beforehand .

Input: $a, b, m, t \mid b \in [o, u]$

1. Calculate the starting point $x_0 = l(b) = mb + t$ for the Newton-Rhapson method.
2. Recursively calculate $x_{n+1} = x_n(2 - x_n b)$
3. Taking the last $x_n \approx \frac{1}{b}$ and multiplying it with a gives us the result $r \approx \frac{a}{b}$

How many iterations are necessary?

Answer: It depends on the interval.
When the interval approaches 0 more iterations are necessary as the value of $\frac{1}{b}$ approaches infinity.
This means more steps to come close to the root.

Computation time?
Doing the division according to these steps is obviously significantly slower than doing it in the traditional way.
Depending on the number of iterations the division can take multiple seconds when done homomorphically.

SQUARE ROOT

Methods

The traditional calculation of the square root is not applicable homomorphically (for the same reasons as with the division)

We also find multiple approaches in literature to calculate the square root differently. We won't discuss them in detail, but we can compare them according to the multiplicate depth:

Method	Multiplicative depth	Limitations
Heron/Raphson	$(d+1)*n$	Start value required
Bakshali	$(2d+4)*n$	Start value required
Wilkes	$3*n$	Range limited to $[0,1)$
Halley	$4*n$	Start value required
Newton-Raphson	$n*3+2$	Start value required

n: Iterations needed for the respective procedure
d: multiplicative depth of the implementation of the required division function

The Newton-Raphson method is the best approach as a shallower multiplicative depth is very important for our use case.

Table taken from Prantl, T., Horn, L., Engel, S. et al. De Bello Homomorphico: Investigation of the extensibility of the OpenFHE library with basic mathematical functions by means of common approaches using the example of the CKKS cryptosystem. Int. J. Inf. Secur. 23, 1149–1169 (2024). https://doi.org/10.1007/s10207-023-00781-0

Square Root with Newton-Raphson

Analog to division we define a function $g(x)$ which has its zero point at $\sqrt{a}$ and use Newton's method to find the zero-point.

Simple choice for $g(x)$:

$$g(x) = x^2 - a$$

This satisfies our requirements as $g(\sqrt{a}) = 0$.

Problem

With $g(x)$, Newton's method gives us this for x_{n+1}:

$$x_{n+1} = x_n - \frac{g(x_n)}{g'(x_n)} = x_n - \frac{x_n^2 - a}{2x_n}$$

This is ok, but we need the division (which adds extra error to our calculation).

Square Root with Newton-Raphson

Idea

We choose $g(x)$ to have its zero point at $\frac{1}{\sqrt{a}}$:

$$g(x) = \frac{1}{x^2} - a$$

This eliminates any division during the Newton iteration (similar as it already did during the implementation of the division itself):

$$x_{n+1} = x_n - \frac{g(x_n)}{g'(x_n)} = x_n - \frac{\frac{1}{x_n^2} - a}{-\frac{2}{x_n^3}} = x_n(1.5 - \frac{a}{2}x_n^2)$$

We need to multiply the result of the Newton iteration with a to receive the result:

$$\frac{1}{\sqrt{a}} * a = \sqrt{a}$$

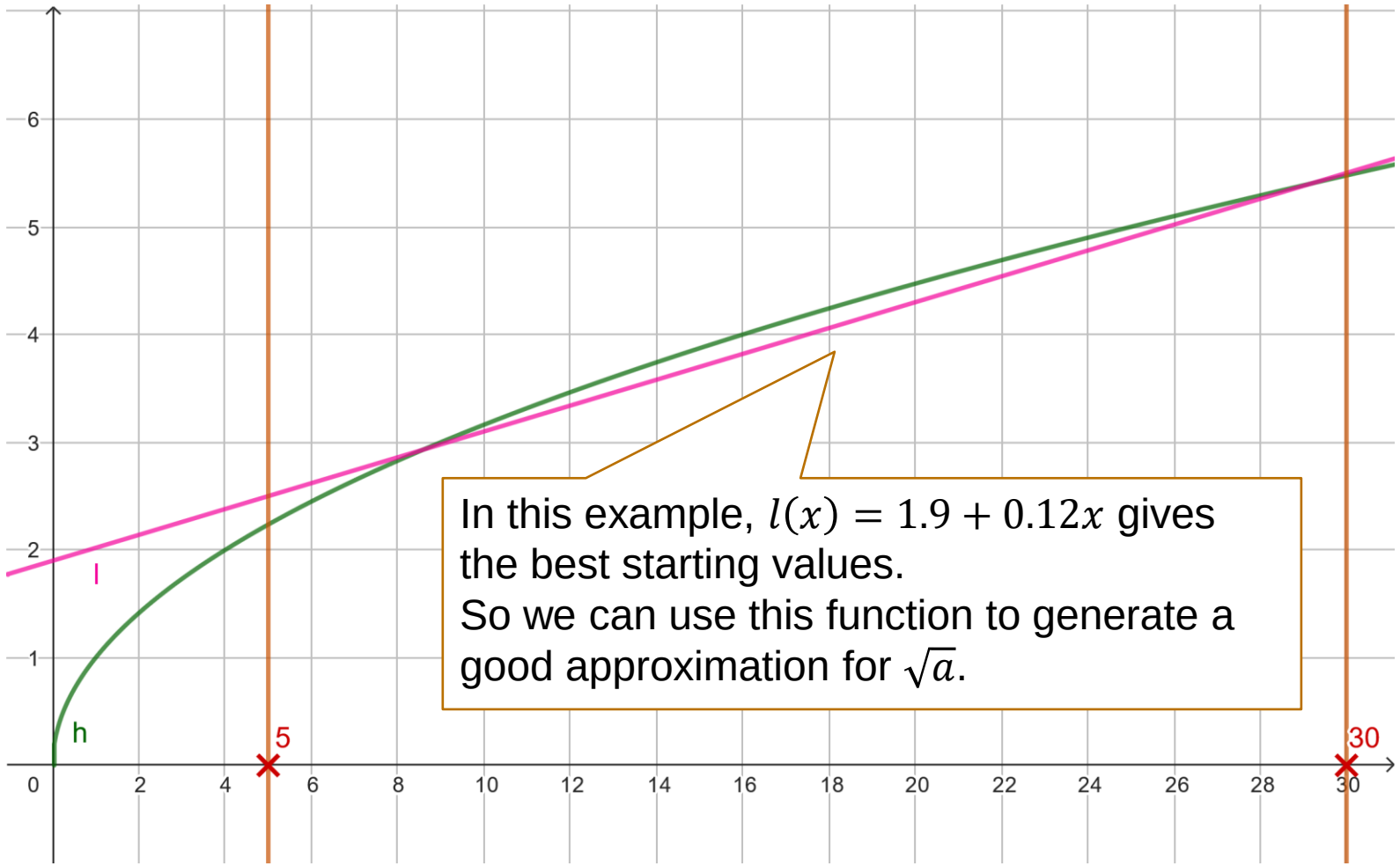
Choosing a starting value

Choosing the starting value can be done the same way as with the division.

We only have a different function to approximate:

$$h(x) = \sqrt{x}$$

By using the range of a we can find a good auxiliary function with brute force.



Square Root – Bringing it all together

To calculate the square root homomorphically, we need these steps:

The parameters for linear approximation are calculated beforehand .

Input: $a, m, t | \sqrt{a} \in [o, u]$

1. Calculate the starting point $x_0 = l(a), l(x) = mx + t$ for the Newton-Rhapson method.
2. Recursively calculate $x_{n+1} = x_n(1.5 - \frac{a}{2}x_n^2)$
3. Taking the last $x_n \approx \frac{1}{\sqrt{a}}$ and multiplying it with a gives us the result $r \approx \sqrt{a}$

How many iterations are necessary?

Answer: It depends on the interval.
For most intervals 10 iterations provide enough precision.

Computation time?
Doing the square root according to these steps is obviously significantly slower, than doing it in the traditional way.
Depending on the number of iterations calculating the square root can take even more than 10 seconds.

EXPONENTIAL FUNCTION & LOGARITHM

Methods

The exponential function and logarithm are traditionally implemented using lookup tables, rounding and algorithms that involve conditional statements.
This is the reason why we also need a different approach when calculating these functions homomorphically.

Note
The following slide shows an overview over different alternative methods found in literature, but we won't discuss them in more detail (as already done with the square root).

Methods

Exponential function

Method	Multiplicative depth	Limitations
Taylor series	$\lceil \log_2(n) \rceil + 1$	
Padé-Approximation	$\lceil \log_2(\max(m,o)) \rceil + d$	Additional coefficients required
Newton-Raphson	$(d + l) * n$	Start value required

n: Iterations needed for the respective procedure
d,l,e,s: multiplicative depth of the implementation of the required division, logarithm, exponential, square root function
m,o: degrees of polynomials used by the Padé-Approximation

Logarithm

Method	Multiplicative depth	Limitations
Modified Taylor series	$\lceil \log_2(2 * n + 1) \rceil + 3 + d$	
Padé-Approximation	$\lceil \log_2(\max(m,o)) \rceil + d$	Additional coefficients required
Newton-Raphson	$(d + e) * n$	Start value required
Arithmetic-geometric Mean	$s + 2d$	Start value required

For both functions we will use the Taylor series as it combines the most advantages!

Tables taken from Prantl, T., Horn, L., Engel, S. et al. De Bello Homomorphico: Investigation of the extensibility of the OpenFHE library with basic mathematical functions by means of common approaches using the example of the CKKS cryptosystem. Int. J. Inf. Secur. 23, 1149–1169 (2024). https://doi.org/10.1007/s10207-023-00781-0

Exponential function using Taylor series

Exponential function and Logarithm are originally defined using a Taylor series

Definition

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

As this definition only requires addition, multiplication and division, we can do it homomorphically.

Optimization

By using the laws $e^{x+y} = e^x * e^y$ we can reduce the number of terms needed from the Taylor series to achieve the same precision.

Logarithm using Taylor series

In literature we find this:

$$\ln\left(\frac{1+x}{1-x}\right) = 2 * \sum_{m=0}^{\infty} \frac{x^{2m+1}}{2m+1}, x \in [-1,1]$$

At first glance, this does not help much, but considering

$$z = \frac{1+\frac{z-1}{z+1}}{1-\frac{z-1}{z+1}}, z \in \mathbb{R}^{+}\backslash\{0\}$$

we can rewrite

$$\ln(z) = \ln\left(\frac{1+x}{1-x}\right) = 2 * \sum_{m=0}^{\infty} \frac{x^{2m+1}}{2m+1}$$

What about the requirement, that $-1 < x < 1$?

with $x = \frac{z-1}{z+1}$.
We can use this to calculate the logarithm of a given z.

Logarithm using Taylor series

We can show that $-1 < x = \frac{z-1}{z+1} < 1$:

$z \in \mathbb{R}^+ \backslash \{0\}$

$$z > 0$$

Subtracting 1

$$-2z < 0$$

$$-2z - 1 < -1$$

Adding z

$$-2z - 1 < -1 < 1$$

$$-z - 1 < z - 1 < z + 1$$

As $z > 0$ we can divide by $(z + 1)$:

$$\frac{-z-1}{z+1} < \frac{z-1}{z+1} < \frac{z+1}{z+1}$$

$$-1 < x < 1$$

This means, that we can use our modified Taylor series to calculate the logarithm for an arbitrary input value.

Logarithm using the Taylor series

Exponential function and Logarithm are originally defined using a Taylor series

Definition

$$\ln(z) = \ln\left(\frac{1+x}{1-x}\right) = 2 * \sum_{m=0}^{\infty} \frac{x^{2m+1}}{2m+1}$$

$$x = \frac{z-1}{z+1}$$

As this definition only requires addition, multiplication and division, we can do it homomorphically.

Optimization

For the logarithm we can rewrite it as $\log(x) = \log\left(\frac{x}{n} * n\right) = \log\left(\frac{x}{n}\right) + \log(n)$. This allows us to precompute $\log(n)$ and only doing $\log\left(\frac{x}{n}\right)$ on the fly (which requires less Taylor terms).

Evaluation of Exponential function & Logarithm with Taylor

How many Taylor terms are necessary to achieve a good precision?

Exponential
The number of Taylor terms needed for a certain precision increases with the input value.
For example: To calculate e^{20} more than 10 terms are needed.

Logarithm
The number of Taylor terms needed for a certain precision also increases with the input value.
But: 8 terms seem to be enough to calculate $\log(x)$ while $x \in]0,200]$

Computation time?

This also depends on the number of Taylor terms.
We see a double-digit number of seconds even for a small number of terms.

MAXIMUM AND MINIMUM

Maximum and Minimum can be implemented directly!

When considering literature, one finds direct implementations of the maximum and minimum function:

$$\max(a,b) = \frac{a+b}{2} + \frac{\sqrt{(a-b)^2}}{2} = 0.5 * (a+b) + 0.5 * \sqrt{(a-b)^2}$$
$$\min(a,b) = a + b - \max(a,b)$$

There are obviously also other approaches (Newton-Raphson, polynomial approximation, …), but these are more complex and need a higher multiplicative depth.

This is why we can stick with this simple approach.

Summary – What did we learn today?

Newton-Raphson
This method can be used to find the roots of a certain function.

Homomorphic implementation of mathematic functions

For **division and square root** we use Newton-Raphson.
Limiting the search area with an interval provides good approximation for the starting value.

For **exponential function and logarithm,** we use their respective definitions as a Taylor series.
Optimizations can be applied

For **maximum and minimum,** we can use a direct definition using division, addition, square root and multiplication

Calculating complex mathematical functions homomorphically is always much slower.
Most of the methods are done iteratively. This means that we only approximate the result of the function.