BOOTSTRAPPING BITS WITH CKKS

Youngjin Bae, Jung Hee Cheon,
Jaehyung Kim and Damien Stehlé

Eprint 2024/767

May 30, 2024

HEAAN CRYPTO LAB

MAIN RESULT

The CKKS FHE scheme is very efficient for binary circuits

$\approx 17\mu s$ per binary gate

(amortized setting, single-thread CPU)

How do we achieve this?

=> With a dedicated bootstrapping algorithm

J. H. Cheon, A. Kim, M. Kim, Y. Song: Homomorphic encryption for arithmetic of approximate numbers. ASIACRYPT'17

CKKS

Cleartexts: vectors in $\mathbb{C}^{N/2}$, with $N \in \{2^{14}, 2^{15}, 2^{16}\}$

Native hom. ops.: add, mult & conj in // (SIMD)
rotation of coordinates

J. H. Cheon, A. Kim, M. Kim, Y. Song: Homomorphic encryption for arithmetic of approximate numbers. ASIACRYPT'17

CKKS

Cleartexts: vectors in $\mathbb{C}^{N/2}$, with $N \in \{2^{14}, 2^{15}, 2^{16}\}$

Native hom. ops.: add, mult & conj in // (SIMD)
rotation of coordinates

It's very efficient: **75 ns / mult** (amortized cost, prec: 22 bits, GPU)

J. H. Cheon, A. Kim, M. Kim, Y. Song: Homomorphic encryption for arithmetic of approximate numbers. ASIACRYPT'17

CKKS

Cleartexts: vectors in $\mathbb{C}^{N/2}$, with $N \in \{2^{14}, 2^{15}, 2^{16}\}$

Native hom. ops.: add, mult & conj in // (SIMD)
rotation of coordinates

It's very efficient: **$75\ ns$ / mult** (amortized cost, prec: 22 bits, GPU)

Why not using it for discrete data?

WHY DISCRETE DATA?

Why discrete data?

- Databases
- Financial transactions
- Blockchain
- Crypto primitives (AES, PRFs, Sigs, etc)

L. Ducas, D. Micciancio: FHEW: Bootstrapping homomorphic encryption in less than a second. EUROCRYPT'15

I. Chillotti, N. Gama, M. Georgieva, M. Izabachène: Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. ASIACRYPT'16

WHY DISCRETE DATA?

Why discrete data?

- Databases
- Financial transactions
- Blockchain
- Crypto primitives (AES, PRFs, Sigs, etc)

Common belief:

- Bit operations => DM/CGGI
- Approx. reals => CKKS

N. Drucker, G. Moshkowich, T. Pelleg, H. Shaul: *BLEACH: cleaning errors in discrete computations over CKKS.* J. Cryptol.'24

BINARY COMPUTATIONS WITH CKKS

DMPS approach

$$b \in \{0,1\} \quad \leftrightarrow \quad b + \varepsilon \in \mathbb{R}$$

(for some $\varepsilon \ll 1$)

N. Drucker, G. Moshkowich, T. Pelleg, H. Shaul: *BLEACH: cleaning errors in discrete computations over CKKS.* J. Cryptol.'24

BINARY COMPUTATIONS WITH CKKS

DMPS approach

$$b \in \{0,1\} \quad \leftrightarrow \quad b + \varepsilon \in \mathbb{R}$$

(for some $\varepsilon \ll 1$)

Format compatible with binary gates

$$\mathrm{G_{OR}}(b_1, b_2) = b_1 + b_2 - b_1 \cdot b_2$$

(over $\mathbb{Z}$)

$$\mathrm{G_{OR}}(b_1 + \varepsilon_1, b_2 + \varepsilon_2) = b_1 + b_2 - b_1 \cdot b_2 + \varepsilon_1 + \varepsilon_2 - b_1\varepsilon_2 - b_2\varepsilon_1 - 2\varepsilon_1\varepsilon_2 \approx b_1 + b_2 - b_1 \cdot b_2$$

(over $\mathbb{R}$)

N. Drucker, G. Moshkowich, T. Pelleg, H. Shaul: *BLEACH: cleaning errors in discrete computations over CKKS.* J. Cryptol.'24

BINARY COMPUTATIONS WITH CKKS

DMPS approach

$$b \in \{0,1\} \quad \leftrightarrow \quad b + \varepsilon \in \mathbb{R}$$

(for some $\varepsilon \ll 1$)

Format compatible with binary gates

$$\mathrm{G_{OR}}(b_1, b_2) = b_1 + b_2 - b_1 \cdot b_2$$

(over $\mathbb{Z}$)

$$\mathrm{G_{OR}}(b_1 + \varepsilon_1, b_2 + \varepsilon_2) = b_1 + b_2 - b_1 \cdot b_2 + \varepsilon_1 + \varepsilon_2 - b_1\varepsilon_2 - b_2\varepsilon_1 - 2\varepsilon_1\varepsilon_2 \approx b_1 + b_2 - b_1 \cdot b_2$$

(over $\mathbb{R}$)

All symmetric binary gates can be implemented with mult depth 1

N. Drucker, G. Moshkowich, T. Pelleg, H. Shaul: *BLEACH: cleaning errors in discrete computations over CKKS.* J. Cryptol.'24

CLEANING THE ERROR

The more we compute, the larger ε in $b + \varepsilon$.

If it grows too much, then b becomes ill-defined.

N. Drucker, G. Moshkowich, T. Pelleg, H. Shaul: *BLEACH: cleaning errors in discrete computations over CKKS.* J. Cryptol.'24

CLEANING THE ERROR

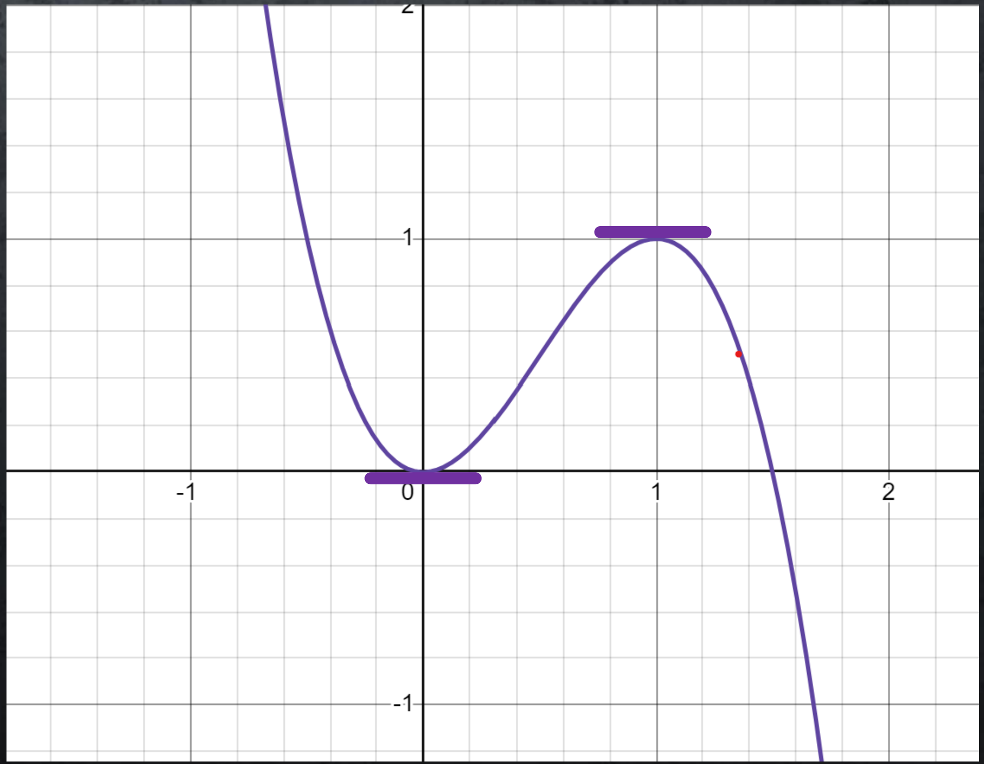
The more we compute, the larger ε in $b+\varepsilon$.

If it grows too much, then b becomes ill-defined.

Noise-cleaning function
$h_1 :\ x \mapsto 3x^2 - 2x^3$

$h_1(0) = 0 \quad h_1(1) = 1$

$h_1'(0) = 0 \quad h_1'(1) = 0$



N. Drucker, G. Moshkowich, T. Pelleg, H. Shaul: *BLEACH: cleaning errors in discrete computations over CKKS.* J. Cryptol.'24

WHAT DO WE DO?

DMPS uses CKKS in a black-box manner

We open the box

We design a **bootstrapping algorithm** for the DMPS format

$\boldsymbol{m} = \boldsymbol{b} + \boldsymbol{\varepsilon} \in \mathbb{C}^{N/2}$, with $\boldsymbol{b} \in \{0,1\}^{N/2}$ and $\boldsymbol{\varepsilon}$ small

J. H. Cheon, K. Han, A. Kim, M. Kim, Y. Song: Bootstrapping for approximate homomorphic encryption. EUROCRYPT'18

THE EVALMOD STEP OF CKKS-BTS

EvalMod: $m + q_0 \cdot I$ for some $I \in \mathbb{Z}$ $\Rightarrow m$

J. H. Cheon, K. Han, A. Kim, M. Kim, Y. Song: Bootstrapping for approximate homomorphic encryption. EUROCRYPT'18

THE EVALMOD STEP OF CKKS-BTS

EvalMod: $m + q_0 \cdot I$ **for some** $I \in \mathbb{Z}$ $\Rightarrow m$

But… we can only evaluate polynomials… and mod-1 isn't even continuous

J. H. Cheon, K. Han, A. Kim, M. Kim, Y. Song: Bootstrapping for approximate homomorphic encryption. EUROCRYPT'18

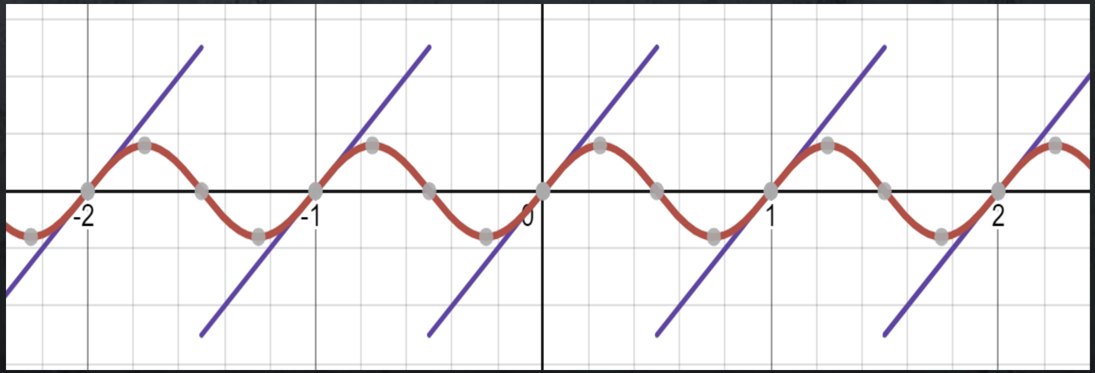
THE EVALMOD STEP OF CKKS-BTS

EvalMod: $m + q_0 \cdot I$ **for some** $I \in \mathbb{Z}$ $\Rightarrow m$

But… we can only evaluate polynomials… and mod-1 isn't even continuous

1- $x \mapsto x \bmod 1$ ⇒ $x \mapsto \frac{1}{2\pi}\sin(2\pi x)$

2- $x \mapsto \frac{1}{2\pi}\sin(2\pi x)$ ⇒ polynomial



BOOTSTRAPPING BITS: MAIN OBSERVATION

EvalMod: $$\frac{1}{q_0} m + I \Rightarrow \frac{1}{q_0} m$$

If m is **real**, mod-1 should be approximated on **wide intervals**

If m is a **bit**, mod-1 can be approximated on **a discrete set**

BOOTSTRAPPING BITS: MAIN OBSERVATION

EvalMod: $$\frac{1}{q_0} m + I \Rightarrow \frac{1}{q_0} m$$

If m is **real**, mod-1 should be approximated on **wide intervals**

~~If m is a **bit**, mod-1 can be approximated on **a discrete set**~~

If m is a **bit**, mod-1 can be approximated on **very narrow intervals** (data is noisy)

BOOTSTRAPPING BITS: MAIN OBSERVATION

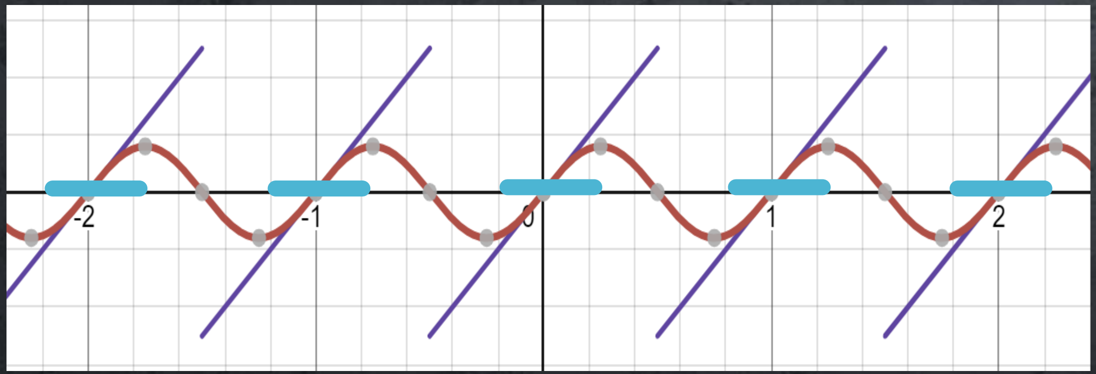
EvalMod: $\frac{1}{q_0} m + I \Rightarrow \frac{1}{q_0} m$

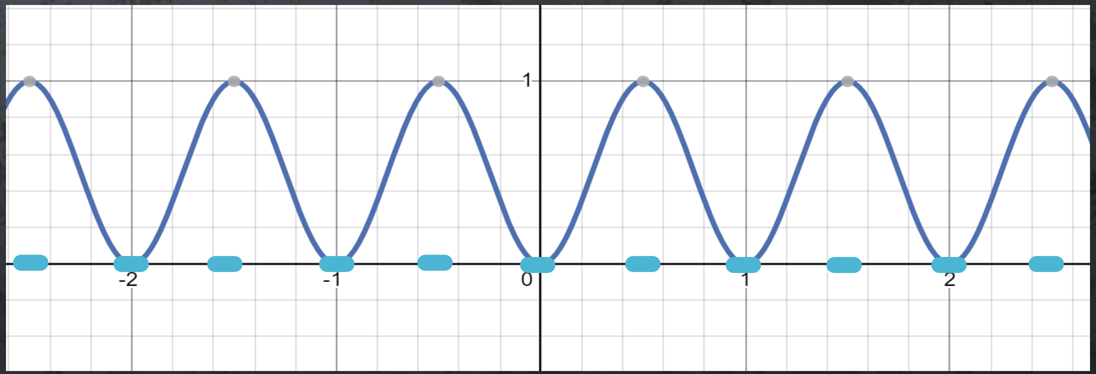
If m is **real**, mod-1 should be approximated on **wide intervals**

~~If m is a **bit**, mod-1 can be approximated on **a discrete set**~~
If m is a **bit**, mod-1 can be approximated on **very narrow intervals** (data is noisy)

Smaller intervals, but twice more: is it easier?

MODIFYING EVALMOD





(Interval of interest)

$$\frac{1}{2\pi}\sin(2\pi x + I) \approx x$$

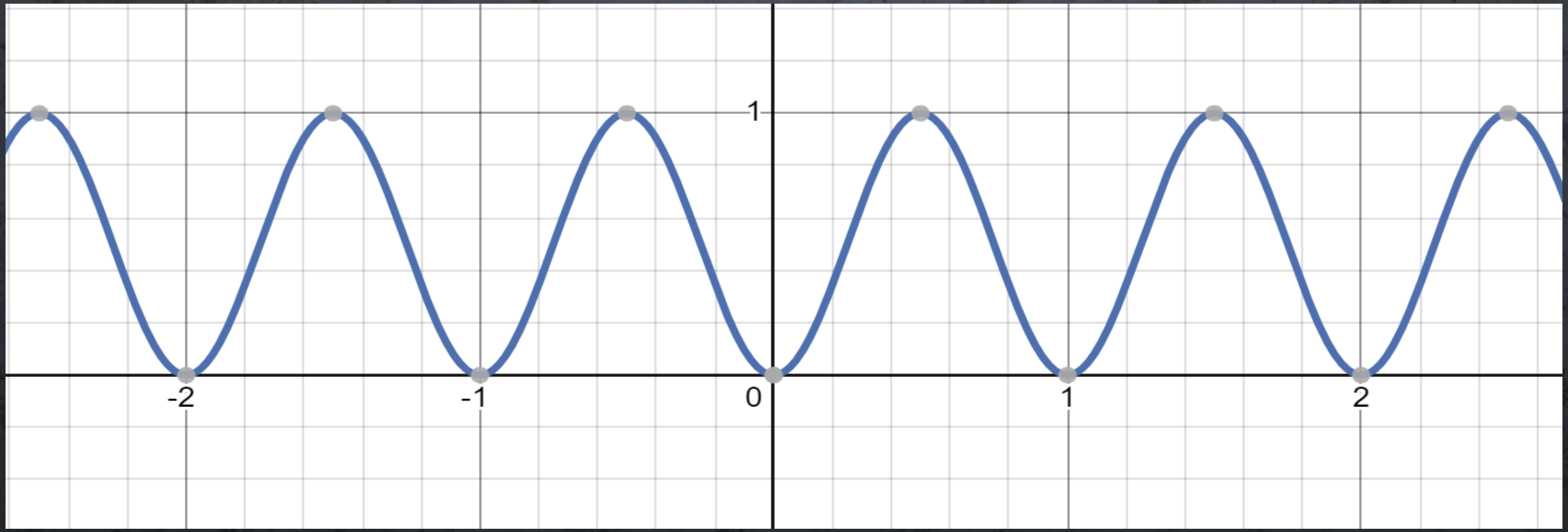
for small x

$$\frac{1}{2}\left(1+\sin\left(2\pi x - \frac{\pi}{2}\right)\right)$$

for $x = \frac{b}{2} + \varepsilon$ and $b \in \{0,1\}$

$\frac{1}{2}\left(1+\sin\left(2\pi x-\frac{\pi}{2}\right)\right)$ for $x=\frac{b}{2}+\varepsilon$

NEW EVALMOD



$\mathbb{Z}$ is mapped to 0

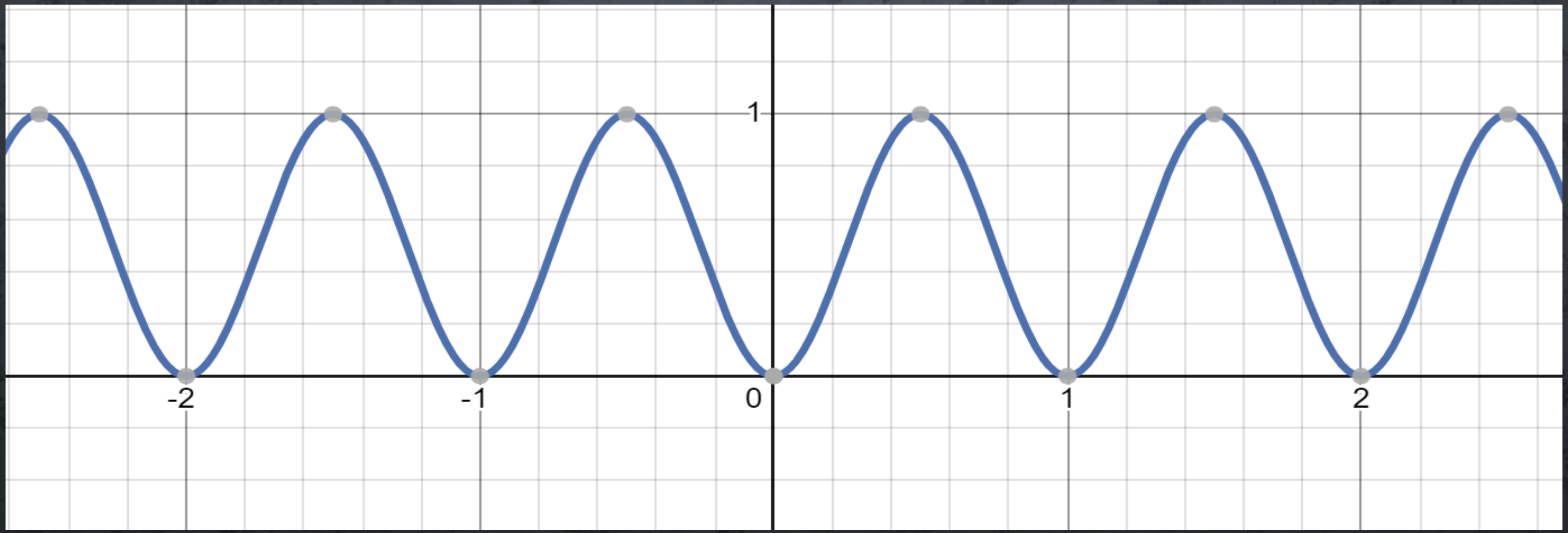
$\mathbb{Z}+\frac{1}{2}$ is mapped to 1

It's correct ☺

It's the almost the same function as before: it's efficient (and easy to implement !!!)

BONUS #1: FREE CLEANING

$\frac{1}{2}\left(1+\sin\left(2\pi x-\frac{\pi}{2}\right)\right)$ for $x=\frac{b}{2}+\varepsilon$



$\mathbb{Z}$ is mapped to 0

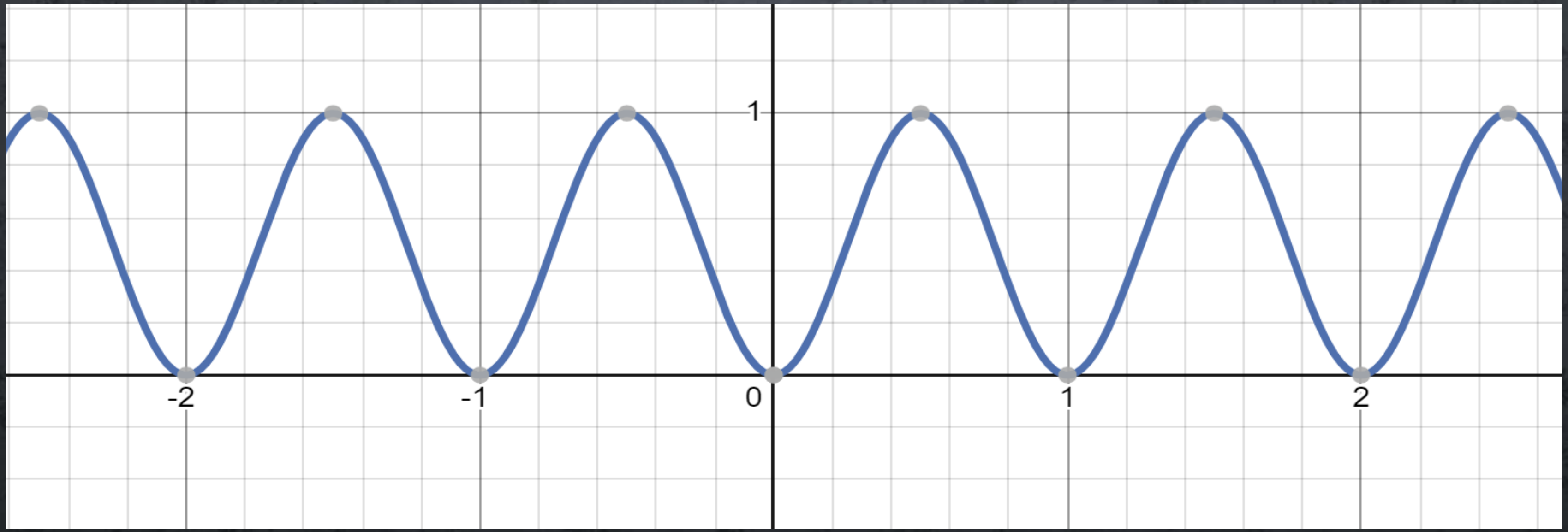
$\mathbb{Z}+\frac{1}{2}$ is mapped to 1

On those points, the derivative is 0

It bootstraps and **cleans** at the same time

$$\frac{b}{2}+\varepsilon \quad \Rightarrow \quad b+O(\varepsilon^2)$$

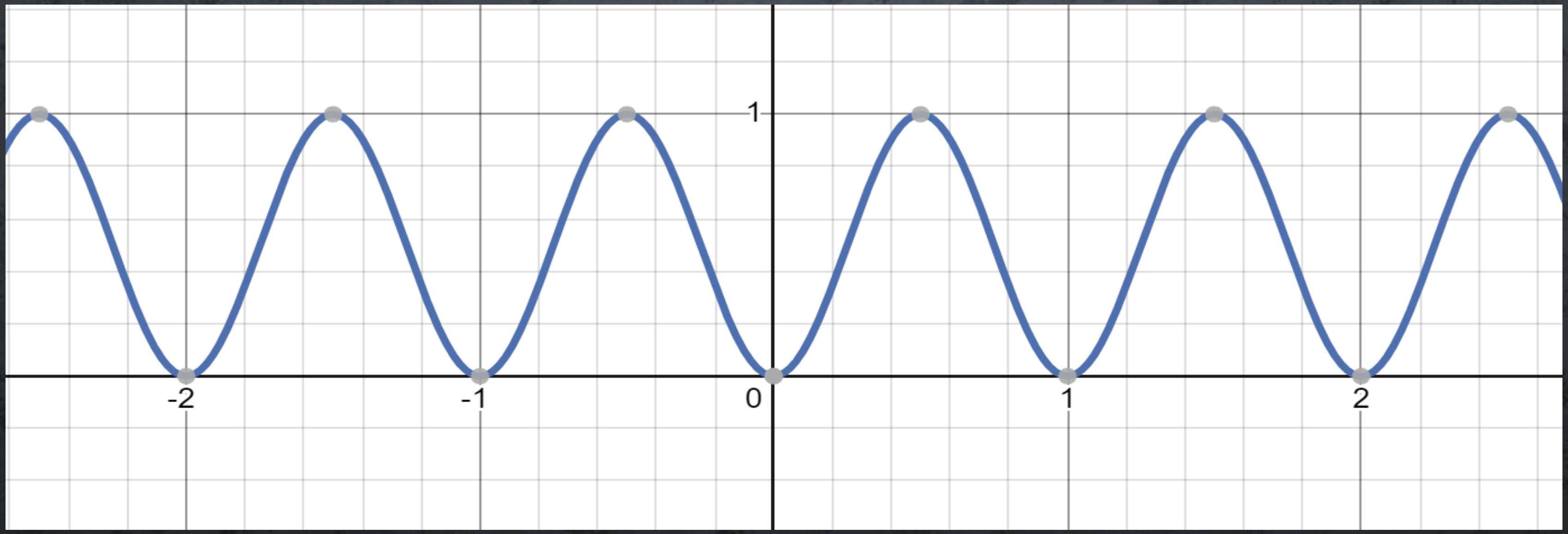
BONUS #2: LOW MODULUS CONSUMPTION



The message is $\frac{b}{2} + \varepsilon + I$ rather than $\frac{1}{q_0} m + I$ with $\frac{1}{q_0} m \ll 1$

$\approx 5 + 5 = 10$ **bits** $\approx 5 + 10 + 5 = 20$ **bits**

BONUS #2: LOW MODULUS CONSUMPTION



The message is $\frac{b}{2} + \varepsilon + I$ rather than $\frac{1}{q_0} m + I$ with $\frac{1}{q_0} m \ll 1$

$\approx 5 + 5 = 10$ **bits** $\approx 5 + 10 + 5 = 20$ **bits**

We decrease the moduli used for BTS,

and **use the freed modulus for more computations**

N. Drucker, G. Moshkowich, T. Pelleg, H. Shaul: *BLEACH: cleaning errors in discrete computations over CKKS.* J. Cryptol.'24

I. Chillotti, N. Gama, M. Georgieva, M. Izabachène: Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. ASIACRYPT'16

EXPERIMENTALLY

	CGGI	DMPS (naive)	DMPS (optimized)	With our BTS
Throughput (amortized time / gate) single-thread CPU	10.5ms	$92.6\mu s$	$27.7\mu s$	$17.6\mu s$

CGGI16: state of art DM/CGGI implementation

DMPS24 naive: clean after every gate (our implementation)

DMPS24 optimized: clean after every few gates (our implementation)

DM/CGGI VS CKKS

	CGGI	Our algorithm
Throughput (amortized time / gate)	10.5ms	$17.6\mu s$
Latency	10.5ms	23.1s

For **heavily parallel** computations, use CKKS

For **non-parallel** computations, use DM/CGGI

What is the limit?

BOOTSTRAPPING DM/CGGI WITH CKKS

Alternative DM/CGGI batch-bootstrapping:

Input: many DM/CGGI ciphertexts

1. Batch the ciphertexts into a single CKKS ciphertext (ring packing)
2. Use our bootstrapping
3. Convert back to DM/CGGI ciphertexts (no cost)

BOOTSTRAPPING DM/CGGI WITH CKKS

Alternative DM/CGGI batch-bootstrapping:

Input: many DM/CGGI ciphertexts

1. Batch the ciphertexts into a single CKKS ciphertext (ring packing)
2. Use our bootstrapping
3. Convert back to DM/CGGI ciphertexts (no cost)

for $\approx$ 170 gates, better use CKKS

(with latency-optimized params)

WRAPPING UP

- CKKS can be used efficiently for **binary circuits**
- For **throughput**: close to **1000x** faster than DM/CGGI
- For **latency**: below ~**170** parallel gates -> DM/CGGI
  above ~**170** parallel gates -> CKKS

QUESTIONS?