Efficient Homomorphic Comparison Methods with **Optimal Complexity**

ASIACRYPT 2020

Jung Hee Cheon, Dongwoo Kim and **Duhyeong Kim**

Seoul National University

This Work

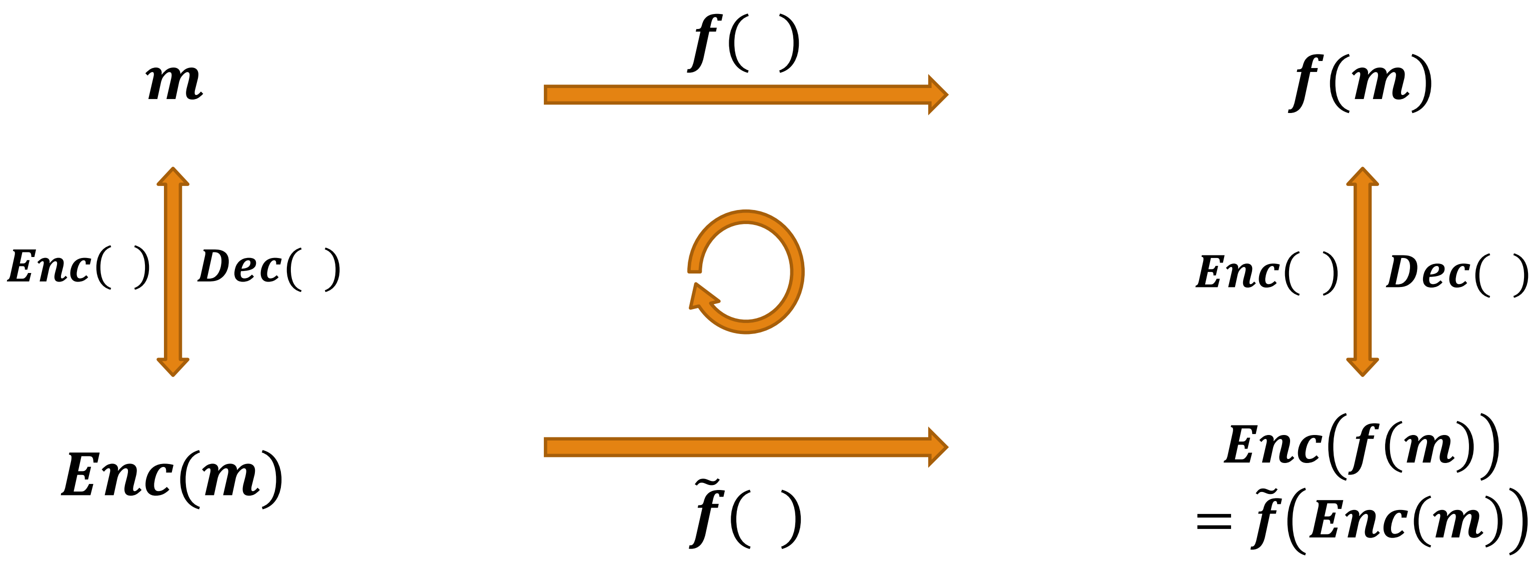
- Complexity-Optimal **Homomorphic Comparison** Method for word-wise HEs
  - Follow-up Study of **[CKK+19]** (Asiacrypt'19)
    - **Quasi-optimal** solution for homomorphic comparison
    - **Impractical** to use (e.g., over 47 minutes for 20-bit integer comparison)
  - **(Optimality)** Requires "asymptotically minimal" homomorphic multiplications
  - **(Practicality)** Comparable to "bit-wise" homomorphic comparison in amortized time
  - **(Mathematical Perspective)** A new framework "composite polynomial approximation" for sign function

[CKK+19] J.H. Cheon, D. Kim, D. Kim et al. "Numerical Methods for Comparison on Homomorphically Encrypted Numbers." **ASIACRYPT 2019**

Backgrounds

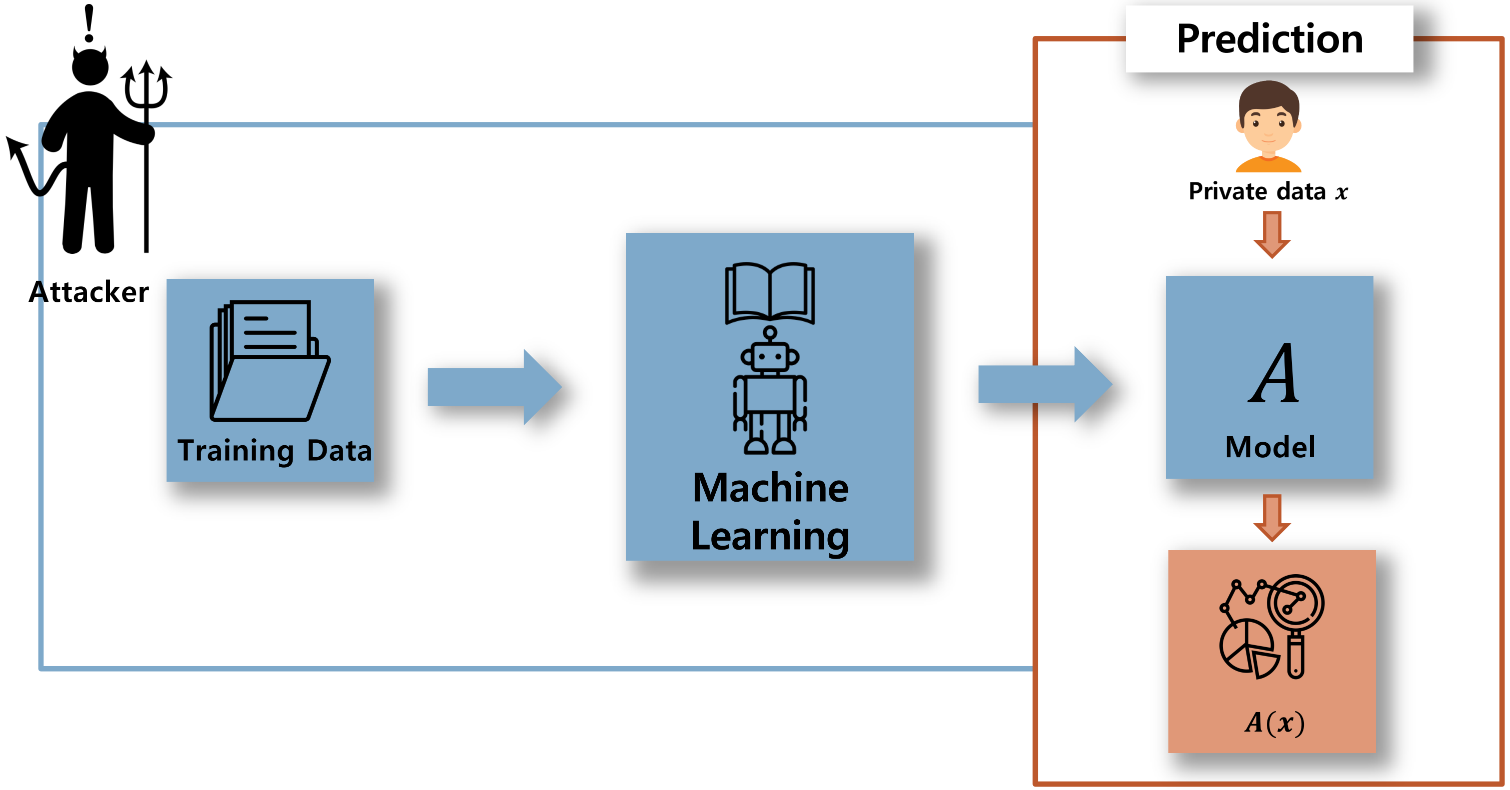
Homomorphic Encryption

- **Homomorphic Encryption (HE)**
  - Allows any computation on encrypted data "without decryption process"



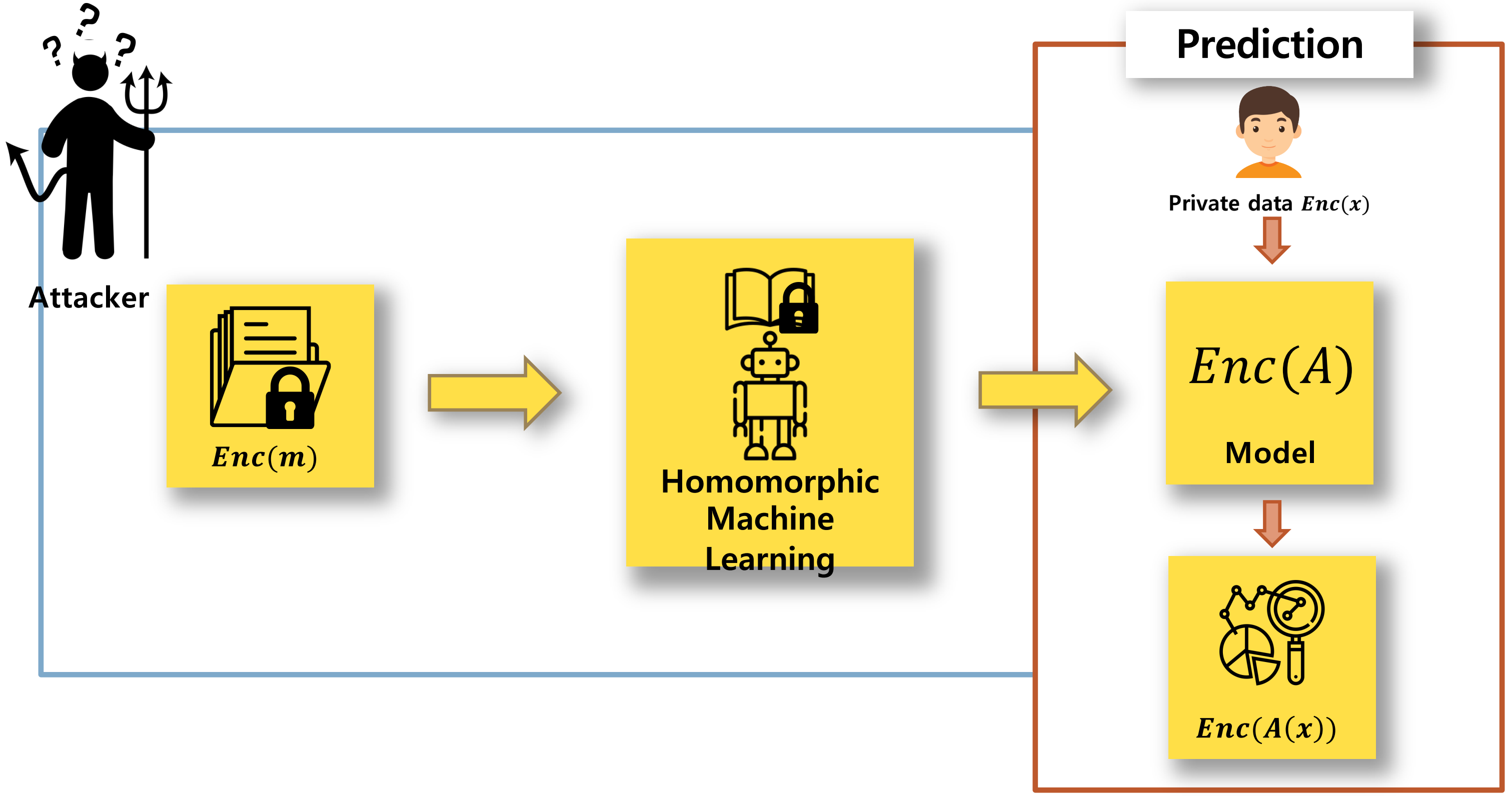
Homomorphic Encryption

- **Ex) Privacy-preserving Machine Learning**



Homomorphic Encryption

- **Ex) Privacy-preserving Machine Learning**



Homomorphic Encryption

Q) What are the limitations of applying HE to real-world applications?

Ans) Computational Inefficiency due to **restricted** basic homomorphic **operations**

Word-wise Approach	Bit-wise Approach
10 -> Enc(10)	10 -> (Enc(1),Enc(0),Enc(1),Enc(0))
Add & Mult easy	**Add & Mult hard**
Comparison hard	Comparison easy
BGV, B/FV, CKKS	FHEW, TFHE, word-wise HE with bit-wise encoding

In this talk, we focus on making up for the weakness of word-wise approach!

Polynomial Approximation

- Imagine that we only have two tools: **Addition** and **Multiplication**
- Then, how we can we evaluate “**non-polynomial**” functions including **comparison**?

$\Longrightarrow$ Approximately compute via **polynomial approximation!**

- Various general Poly. Approx. methods in numerical analysis
  - Taylor (local), Least square approximation (L2-norm), minimax ($L\infty$-norm), Chebyshev, etc.

- Due to these well-studied Poly. Approx. methods, one may think we’ve already done (?)
  - Theoretically, we may say…yes
  - But in efficiency and practicality, hmm…long way to go!

Polynomial Approximation

- **Limitations** of general polynomial approximation methods
  - Aim to find the relation between **degree** and **error bound**
  - They output **"minimal-degree"** polynomial within a certain error bound under some error measure
  - BUT, the number of multiplications (**complexity**) is also an very important factor, more critical in HE

"Can we find a new polynomial approximation method (for the sign function) with minimal complexity rather than degree?"

High-level Idea

High-level Idea [CKK+19, this work]

- To approximate a non-polynomial function with some **"structured"** polynomials
  - An "unstructured" poly G requires at least $\Theta\left(\sqrt{\deg G}\right)$ multiplications [PS73]
    - For $|x|$, to obtain α-bit precision output via minimax poly. Approx. over [-1,1], $\Theta\left(2^{\alpha/2}\right)$ multiplications are required
  - For $\boldsymbol{F = f \circ f \circ \cdots \circ f}$ for a const-degree f, then it requires only $\Theta(\log \deg F)$ multiplications
  - If $\deg F = o(2^{\deg G})$, then F evaluation requires (asymptotically) **less complexity** than G evaluation.

[CKK+19] J.H. Cheon, D. Kim, D. Kim et al. "Numerical Methods for Comparison on Homomorphically Encrypted Numbers." **ASIACRYPT 2019**

[PS73] Paterson, Michael S., and Larry J. Stockmeyer. "On the number of nonscalar multiplications necessary to evaluate polynomials." *SIAM Journal on Computing* 2.1 (1973): 60-66.

High-level Idea [CKK+19, this work]

The previous work **[CKK+19]** **finds** such structured polynomials **from the literature of numerical analysis**

In this work, we aim to construct a new framework for composite polynomial approximation,

rather than exploiting existing algorithms

Go Into Detail

Previous Work [CKK+19]

- **Main Idea**
  - Composite Polynomial $\Leftrightarrow$ **"Iterative Algorithm"**
  - Express the comparison function as a rational function:

$$\text{Comp}(a,b) = \begin{cases} 1 & if\ a > b \\ \frac{1}{2} & if\ a = b \\ 0 & if\ a < b \end{cases} = \lim_{d\to\infty} \frac{a^{2^d}}{a^{2^d}+b^{2^d}}$$

Goldschmidt's **iterative** algorithm for **"division"** [Gol64]

  - More specifically, $\frac{a^{2^d}}{a^{2^d}+b^{2^d}}$ is evaluated by iterative computations of $a \leftarrow \frac{a^2}{a^2+b^2}$ and $b \leftarrow \frac{b^2}{a^2+b^2}$

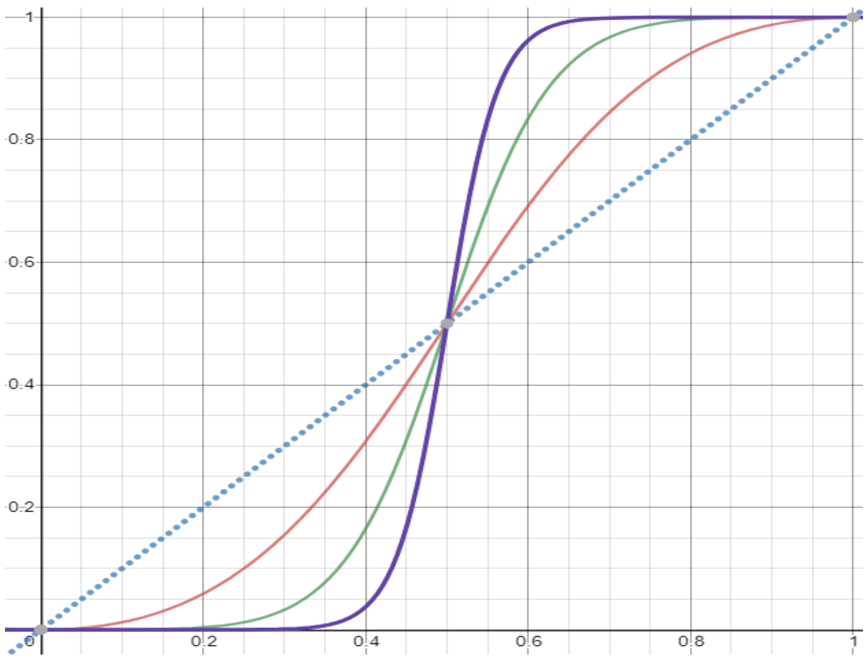
[Gol64] Goldschmidt, R.E. Applications of division by convergence. Ph.D. thesis, Massachusetts Institute of Technology (1964)

Our Work

- **Key Observation**
  - The previous approach can be interpreted as the following two steps
    1. Normalize inputs $a \leftarrow \frac{a}{a+b}$ and $b \leftarrow \frac{b}{a+b}$ so that $a + b = 1$
    2. Iteratively compute a rational function $a \leftarrow f_0(a) = \frac{a^2}{a^2+b^2} = \frac{a^2}{a^2+(1-a)^2}$
  - **Re-interpret**: $f_0^{(d)} = f_0 \circ f_0 \circ f_0 \circ \cdots \circ f_0$ gets close to $\chi_{\left(\frac{1}{2},\infty\right)}(x) = \frac{\mathrm{sgn}(2x-1)+1}{2}$ over [0,1] as $d \leftarrow \infty$

Our Work

- The graph represents $f_0^{(d)}$ for $d = 1,2,3$



Our Work

- **Key Observation**
  - The basic function f does **NOT** need to be the rational function $f_0(x) = \frac{x^2}{x^2+(1-x)^2}$ which contains **expensive division** operation
  - Instead, symmetry w.r.t. (1/2,1/2), convexity, and some other things may be enough

"What are the core properties of f which makes $f^{(d)}$ get close to the sign function?"

"Equivalence": $\chi_{\left(\frac{1}{2},\infty\right)} = \mathrm{sgn} = comp$

$$comp(a,b) = \frac{\mathrm{sgn}(a-b)+1}{2}$$

Our Work

- **Core properties of f:**
  1. $f(-x) = -f(x)$ (Origin Symmetry)
  2. $f(1) = 1$ (Convergence to ± 1)
  3. $f'(x) = c(1-x^2)^n$ for some $c > 0$ (Faster Convergence; Optional)

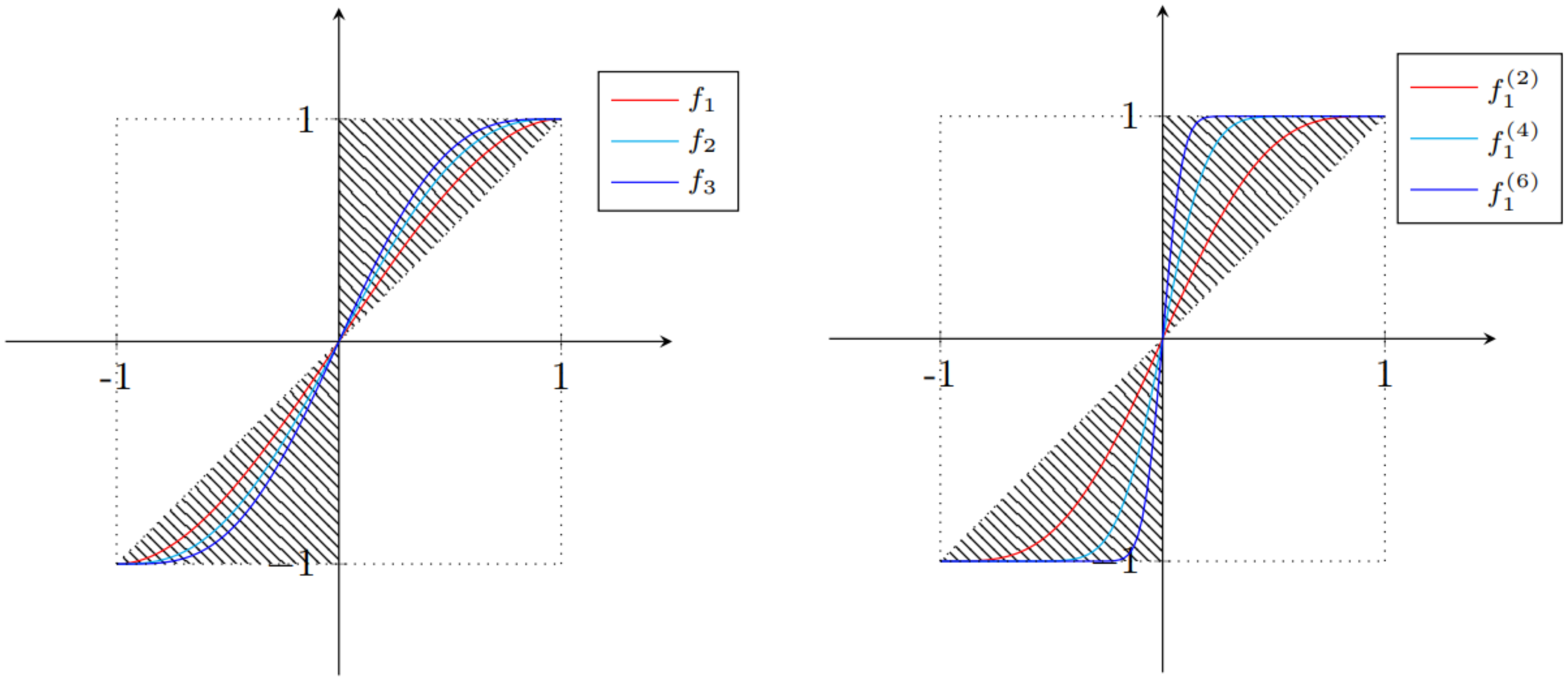
- Such f is "uniquely" determined for each n:

$$f_n(x) = \sum_{i=0}^{n} \frac{1}{4^i} \cdot \binom{2i}{i} \cdot x(1-x^2)^i$$

  - $f_1(x) = -\frac{1}{2}x^3 + \frac{3}{2}x$
  - $f_2(x) = \frac{3}{8}x^5 - \frac{10}{8}x^3 + \frac{15}{8}x$

Our Work

- Graphs of $f_n^{(d)}$ for various n and d



(a) f_n for $n = 1, 2, 3$

(b) $f_1^{(d)}$ for $d = 2, 4, 6$

Our Work

Theorem 1. If the number of compositions $d \geq \frac{1}{\log f_n'(0)} \cdot \log\left(\frac{1}{\epsilon}\right) + \frac{1}{\log(n+1)} \cdot \log \alpha + O(1)$,

then it holds that $\left\| f_n^{(d)}(x) - \mathrm{sgn}(x) \right\| \leq 2^{-\alpha}$ for $x \in [-1, -\epsilon] \cup [\epsilon, 1]$.

Our Work

Theorem 1. If the number of compositions $d \geq \frac{1}{\log f_n'(0)} \cdot \log\left(\frac{1}{\epsilon}\right) + \frac{1}{\log(n+1)} \cdot \log \alpha + O(1)$

then it holds that $\left\| f_n^{(d)}(x) - \mathrm{sgn}(x) \right\| \leq 2^{-\alpha}$ for $x \in [-1, -\epsilon] \cup [\epsilon, 1]$.

<The goal of the composition>

To put $[\boldsymbol{\epsilon}, \mathbf{1}]$ into $[\mathbf{1} - \mathbf{2}^{-\boldsymbol{\alpha}}, \mathbf{1}]$

(and $[-\mathbf{1}, -\boldsymbol{\epsilon}]$ into $[-\mathbf{1}, -\mathbf{1} + \mathbf{2}^{-\boldsymbol{\alpha}}]$)

Our Work

Theorem 1. If the number of compositions $d \geq \frac{1}{\log f_n'(0)} \cdot \log\left(\frac{1}{\epsilon}\right) + \frac{1}{\log(n+1)} \cdot \log \alpha + O(1)$,

then it holds that $\left\| f_n^{(d)}(x) - \mathrm{sgn}(x) \right\| \leq 2^{-\alpha}$ for $x \in [-1, -\epsilon] \cup [\epsilon, 1]$.

Put $[\epsilon, 1]$ into $[1 - c, 1]$

Put $[1 - c, 1]$ into $[1 - 2^{-\alpha}, 1]$

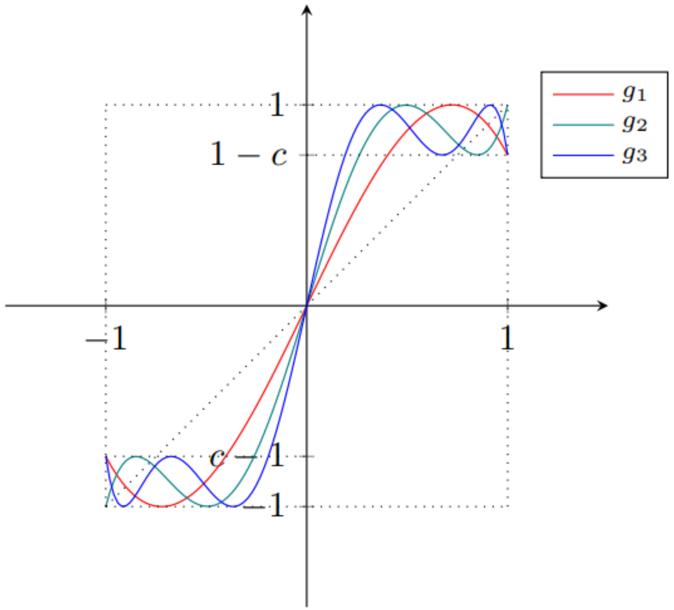
- Core Property 2 and 3 of f_n
  - Adequate for the second goal $[1 - c, 1] \Longrightarrow [1 - 2^{-\alpha}, 1]$
  - But, **NOT** necessary for the **first goal** $[\epsilon, 1] \Longrightarrow [1 - c, 1]$

Our Work

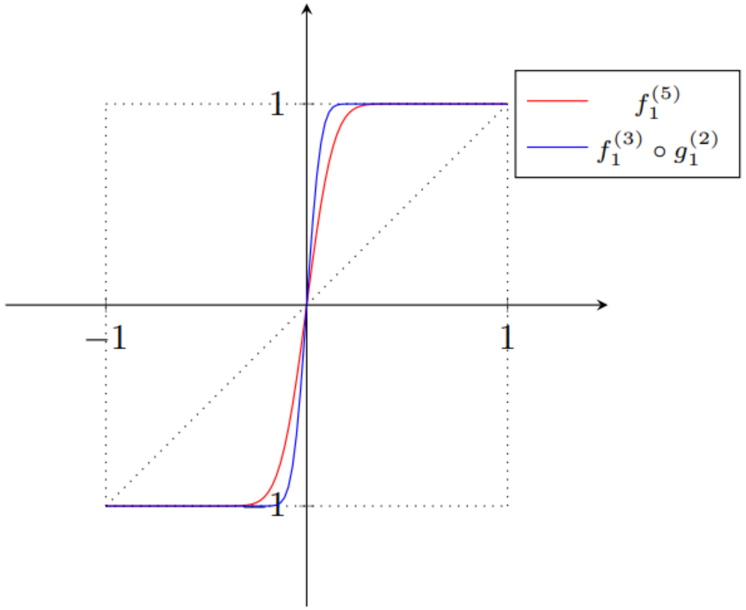
- g **Acceleration method**
  - Find g_n **optimal to the first goal**, and then replace $f_n^{(d)}$ by $f_n^{(d_2)} \circ g_n^{(d_1)}(x) \approx \mathrm{sgn}(x)$ over $[-1,1]$
  - Replace core property 2 and 3 by a **new core property 4** for g_n
  - g_n is **much steeper** than f_n at zero $(g_n'(0) \approx f_n'(0)^2)$ but **not flat** at ± 1

Our Work

- g **Acceleration method**



(a) g_n for $n = 1, 2, 3$ when $c = 1/4$



(b) $f_1^{(5)}$ and $f_1^{(3)} \circ g_1^{(2)}$

Results

- **(Theoretic)** New homomorphic comparison algorithms with **optimal asymptotic complexity**

Parameters	Minimax Approx.	[CKK+19] Method	Our Methods
$\log(1/\epsilon) = \Theta(1)$	$\Theta(\sqrt{\alpha})$	$\Theta(\log^2 \alpha)$	$\boldsymbol{\Theta(\log \alpha)}$
$\log(1/\epsilon) = \Theta(\alpha)$	$\Theta(\sqrt{\alpha} \cdot 2^{\alpha/2})$	$\Theta(\alpha \cdot \log \alpha)$	$\boldsymbol{\Theta(\alpha)}$
$\log(1/\epsilon) = \Theta(2^\alpha)$	$\Theta(\sqrt{\alpha} \cdot 2^{2^{\alpha-1}})$	$\Theta(\alpha \cdot 2^\alpha)$	$\boldsymbol{\Theta(2^\alpha)}$

Results

- **(Practical)** Much faster than the previous **[CKK+19]** method in practice
  - **30 times faster** for the comparison of two 20-bit encrypted integers (with 20-bit output precision)

Precision bits	[CKK+19] method	Our method 1	Our method 2
8	238 s (3.63 ms)*	**59 s (0.90 ms)**	**31 s (0.47 ms)**
12	572 s (8.73 ms)*	**93 s (1.42 ms)**	**47 s (0.72 ms)**
16	1429 s (21.8 ms)*	**151 s (2.30 ms)***	**80 s (1.22 ms)**
20	2790 s (42.6 ms)*	**285 s (4.35 ms)***	**94 s (1.43 ms)***

Implementation based on HEaaN with $N = 2^{17}$ and $h = 256$

An asterisk(*) means that the HEaaN parameter does not achieve 128-bit security due to large $\log Q \geq 1700$

Results

- **(Practical)** Much faster than the previous **[CKK+19]** method in practice
  - **30 times faster** for the comparison of two 20-bit encrypted integers (with 20-bit output precision)

Precision bits	[CKK+19] method	Our method 1	Our method 2
8	238 s (3.63 ms)*	**59 s (0.90 ms)**	**31 s (0.47 ms)**
12	572 s (8.73 ms)*	**93 s (1.42 ms)**	**47 s (0.72 ms)**
16	1429 s (21.8 ms)*	**151 s (2.30 ms)***	**80 s (1.22 ms)**
20	2790 s (42.6 ms)*	**285 s (4.35 ms)***	**94 s (1.43 ms)***

4~10 times faster

2~3 times faster

Further Works & Open Questions

- Follow-up study of this work
  - What is the **"best choice" of $\boldsymbol{n}$**?
    - ✓ In terms of computational complexity, $n = 4$ is the best
    - ✓ Then how about in terms of the **various HE cost models**? (Can we classify the HE cost models?)
  - Proofs for heuristic properties of g acceleration methods
- In general,
  - Can we design new homomorphic comparison algorithms from outside of polynomial evaluation framework?
  - Can we construct a new HE scheme which supports add, mult and comparison as basic operations?

thank you!