Revisiting the Functional Bootstrap in TFHE

Antonio Guimarães\textsuperscript{1}  Edson Borin\textsuperscript{1}  Diego F. Aranha\textsuperscript{2}

September 14, 2021

\textsuperscript{1}Institute of Computing, University of Campinas, Brazil

\textsuperscript{2}Department of Computer Science, Aarhus University, Denmark
Efficiently evaluating non-linear functions with high precision is a challenge for FHE schemes.

CKKS\textsuperscript{1}:
- Approximations using Taylor, Fourier, and Chebyshev series.
- Good performance (SIMD-like computation) for low-precision approximations.

TFHE\textsuperscript{2}:
- Circuits are implemented using binary logic gates.
- Low throughput of operations.
- New approach: \textbf{Functional Bootstrap}\textsuperscript{3}

\textsuperscript{1}Cheon et al., “Homomorphic Encryption for Arithmetic of Approximate Numbers”, 2017.
\textsuperscript{2}Chillotti et al., “TFHE: fast fully homomorphic encryption over the torus”, 2020.
\textsuperscript{3}Boura et al., “Simulating Homomorphic Evaluation of Deep Learning Predictions”, 2019.
Outline

1. TFHE
2. The Functional Bootstrap
3. This work
4. Results
TFHE
TFHE - Fully homomorphic encryption over the Torus

• Security based on the Learning With Errors (LWE) problem.

• The Real Torus:
  - $\mathbb{T} = \mathbb{R}/\mathbb{Z}$: the set of real numbers modulo 1.
  - Unsigned: $[0, 1)$
  - Signed: $[-0.5, 0.5)$

• Ciphertexts:
  - TLWE
  - TRLWE
  - TGSW

---

$^4$Regev, “On Lattices, Learning with Errors, Random Linear Codes, and Cryptography”, 2009.

$^5$Chillotti et al., “TFHE: fast fully homomorphic encryption over the torus”, 2020.
**Definition**

A pair \((a, b) \in \mathbb{T}^{n+1}\), where \(b = \langle a, s \rangle + e\). The vector \(a\) is uniformly sampled from \(\mathbb{T}^n\), the secret key \(s\) is uniformly sampled from \(\mathbb{B}^n\), the error \(e \in \mathbb{T}\) is sampled from a Gaussian distribution with mean 0 and standard deviation \(\sigma\), and \(\langle , \rangle\) denotes the inner product.

- Example (\(n = 5\)):
  - \(a \leftarrow 0.32 \ -0.41 \ -0.12 \ 0.19 \ -0.40\)
  - \(b = \langle a, s \rangle + e\)
  - \(s \leftarrow 1 \ 0 \ 1 \ 0 \ 0\)
  - \(e \leftarrow 0.03\)
  - \(b = \langle a, s \rangle + e = 0.32 + -0.12 + 0.03 = 0.23\)
  - \(c_0 = (a, b) \in \text{TLWE}_s(0)\)
TLWE - Encryption

- Fresh sample of 0:
  - $a \leftarrow \begin{bmatrix} 0.32 & -0.41 & -0.12 & 0.19 & -0.40 \end{bmatrix}$
  - $s \leftarrow \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \end{bmatrix}$
  - $e \leftarrow 0.03$
  - $b = \langle a, s \rangle + e = 0.32 - 0.12 + 0.03 = 0.23$
  - $c_0 = (a, b) \in \text{TLWE}_s(0)$

- Encrypting the message $m = 0.2$:
  - $c_m = c_0 + (0, m) = (a + 0, b + m) = (a, 0.43)$
  - $c_m \in \text{TLWE}_s(m)$

- Decryption:
  - Phase (message + error): $\phi(c_m) = b - \langle a, s \rangle = 0.43 - 0.20 = 0.23$
  - Approximate computing: $m \approx 0.23$
  - Exact computing: $m = \left\lceil \phi(c_m) \right\rceil_{\frac{1}{10}} = 0.2$
TLWE and TRLWE Samples

- TLWE Sample
  - Each element is a scalar in the Torus.

- TRLWE Sample
  - Each element is a polynomial with coefficient in the Torus.
  - Example ($N = 4$):
    - $m \leftarrow 0.32x^3 - 0.41x^2 - 0.12x^1 + 0.19$
Arithmetic

- Additions and multiplications by cleartext
  - \( c_1 + c_2 = (a_1 + a_2, b_1 + b_2) \)
  - \( c_1 \cdot z = (a_1 \cdot z, b_1 \cdot z) \), for \( z \in \mathbb{Z} \).

- Multiplication between ciphertexts
  - TFHE relies on external products.

- The error increases with arithmetic operations
  - Eventually it would affect bits of the message.
  - Reset the error using a Bootstrap.
1. Key Switching
   - Ex: 4 TLWE samples $\text{TLWE}_s(m_i) \rightarrow \text{TRLWE}_S(\sum_{i=0}^{3} m_i X^i)$

2. Sample Extract
   - Ex: SampleExtract$_0$(TRLWE$_S(\sum_{i=0}^{3} m_i X^i)) \rightarrow \text{TLWE}_s(m_0)$
3. Blind Rotate

- $c, \text{ACC} \rightarrow \text{TRLWE}_S(\text{ACC} \cdot X^{[\phi(c)2^N]} \mod \Phi_{2^N})$
- **Input:**
  \[
  \text{ACC} = \text{TRLWE}_S(m_3X^3 + m_2X^2 + m_1X^1 + m_0)
  \]
  
  $c = \text{TLWE}_S(m), m = 0.25$

- **Output:**
  \[
  \text{TRLWE}_S(m_1X^3 + m_0X^2 - m_3X^1 - m_2)
  \]
Bootstrapping

- Boolean values in TFHE: $(0, 1) \mapsto (-\frac{1}{4}, \frac{1}{4})$

- Input:
  - $c \in \text{TLWE}_s(m)$, e.g. $\phi(c) = \frac{1}{8}$
  - $\text{ACC} = \sum_{i=0}^{N} \frac{1}{4}X^i$

- Bootstrap:
  1. Use BlindRotate to calculate $\text{ACC} \cdot X^{\lceil -\phi(c)2^N \rceil \mod \Phi_{2^N}}$
  2. Use SampleExtract to extract the constant term of rotated ACC.

- Result: $\overline{c} \in \begin{cases} 
\text{TLWE}(\frac{1}{4}), & \text{if } \phi(c) \in [0, 0.5), \\
\text{TLWE}(-\frac{1}{4}), & \text{if } \phi(c) \in [-0.5, 0). 
\end{cases}$
The Functional Bootstrap
Functional Bootstrap

Evaluate Lookup Tables (LUTs)

Example
Functional Bootstrap

- Integer values (base $B = 4$): $(0, 1, 2, 3) \mapsto (0, \frac{1}{8}, \frac{2}{8}, \frac{3}{8})$

- ACC now encodes a LUT that evaluates a function $F$:

$$L = [l_1, l_2, l_3, l_4] \mapsto \sum_{i=0}^{255} \frac{l_1}{8} X^i + \sum_{i=256}^{511} \frac{l_2}{8} X^i + \sum_{i=512}^{767} \frac{l_3}{8} X^i + \sum_{i=768}^{1023} \frac{l_4}{8} X^i$$

- $c \in \text{TLWE}_s(m)$ is the LUT selector.

- The bootstrap algorithm is similar:
  1. Add a precision offset to $c$: $c \leftarrow c + (0, 1/16)$.
  2. Use BlindRotate to calculate $\text{ACC} \cdot X[H(c)] \mod \Phi_{2N}$
  3. Use SampleExtract to extract the constant term of rotated ACC.

- Result: $\overline{c} \in \text{TLWE}(F(m))$
Multi-value Bootstrap

• A technique for evaluating multiple LUTs with the same selector.
• Asymptotic cost per LUT: $Base \times N$
• Output error increment$^6$: $\|TV_f\|_2^2 \leq s(q - 1)^2$ times.
  • $s$: input base.
  • $q$: output base.

---

$^6$Sergiu Carpov, Malika Izabachène, and Victor Mollimard. “New Techniques for Multi-value Input Homomorphic Evaluation and Applications”. In: Topics in Cryptology – CT-RSA 2019. Ed. by Mitsuru Matsui. Cham: Springer International Publishing, 2019, pp. 106–126. ISBN: 978-3-030-12612-4. DOI: 10.1007/978-3-030-12612-4\_6.
Evaluating functions with high precision

Table 1: Functional bootstrap performance

|                                   | Precision (bits) | N     | Error Rate ($\log_2$) | Execution Time |
|-----------------------------------|------------------|-------|------------------------|----------------|
| Sign Function$^7$                 | 1                | 1024  | Negligible             | 13 ms          |
| 6-bit-to-6-bit LUT$^9$             | 6                | 16384 | -26.94                 | $\approx 1$ s   |

$^7$Bourse et al., “Fast Homomorphic Evaluation of Deep Discretized Neural Networks”, 2018.
$^8$Izabachène, Sirdey, and Zuber, “Practical Fully Homomorphic Encryption for Fully Masked Neural Networks”, 2019.
$^9$Carpov, Izabachène, and Mollimard, “New Techniques for Multi-value Input Homomorphic Evaluation and Applications”, 2019.
Evaluating functions with high precision

| Function                      | Precision | N     | Error Rate ($\log_2$) | Execution Time (ms) |
|-------------------------------|-----------|-------|-----------------------|---------------------|
| Sign Function$^{10,11}$       | 1         | 1024  | Negligible            | 13 ms               |
| 6-bit-to-6-bit LUT$^{12}$     | 6         | 16384 | -26.94                | $\approx$ 1 s       |
| **6-bit-to-6-bit LUT (This work)** | 6         | 1024  | -59.59                | 378 ms              |

$^{10}$Bourse et al., “Fast Homomorphic Evaluation of Deep Discretized Neural Networks”, 2018.
$^{11}$Izabachène, Sirdey, and Zuber, “Practical Fully Homomorphic Encryption for Fully Masked Neural Networks”, 2019.
$^{12}$Carpov, Izabachène, and Mollimard, “New Techniques for Multi-value Input Homomorphic Evaluation and Applications”, 2019.
This work
Contributions

- Combining Multiple Functional Bootstraps
  - Tree-based Approach
  - Chaining Approach
- Building blocks optimizations
  - Base-aware key switching
  - Multi-value Extract
- Error variance analysis
- Implementation of common functions
  - Up to 8 times speedup over works using logic gates.
  - Up to 3 times speedup over previous works using TFHE’s functional bootstrap.
Combining Multiple Functional Bootstraps

Tree-based method

Chaining method

\[ \bar{\tau} + \frac{1}{2} \] (output)
Chaining Method

**Supported Functions (Recursive definition)**

Let $f$ be a function over an operand $x$ with $d$ digits. For each digit $x_i$ for $i \in [0, d)$, there shall exist a linear combination $\otimes_i$ and a LUT $L_i$, s. t.:

$$
\bar{x}_i = \begin{cases} 
L_i(x_i) = \otimes_f(f(x_i)), & \text{if } i = 0 \\
L_i(\otimes_{i-1}(\bar{x}_{i-1}, x_i)) = \otimes_f(f(x_i : x_0)), & \text{if } i \geq 1.
\end{cases}
$$

- **Suitable Functions**
  - Test logic: integer comparison\(^{13}\), sign, parity.
  - Carry-like logic: addition, multiplication.

---

\(^{13}\)Bourse, Sanders, and Traoré, “Improved Secure Integer Comparison via Homomorphic Encryption”, 2020.
Tree-Based Method

(a) Full-tree

(b) Optimized tree
Tree-based approach

Algorithm 0: Tree-based method for combining functional bootstraps

Input : a set of TLWE samples $c_i \in \text{TLWE}_s\left(\frac{m_i}{2B}\right)$, such that $\sum_{i=0}^{d-1} m_i B^i = m$ is the integer $m$ in base $B$ with $d$ digits.

Input : a set $L$ of $B^d$ polynomials $\in \mathbb{Z}_N[X]$ encoding the lookup table of an arbitrary function $F$.

Input : a bootstrapping key $B_{K_i}$ and a Key Switching key $K_{S_{i,j}}$.

Output: A TLWE sample $\bar{c} \in \text{TLWE}_{\mathbb{S}}\left(\frac{F(m)}{2B}\right)$.

1. $TV \leftarrow L, f : T^B \mapsto T_N[X] = (a_1, ..., a_B) \mapsto a_1 X^{N-1} + ... + a_B$
2. for $i \leftarrow 0$ to $d - 1$ do
3.    $\bar{c} \leftarrow \text{MultiValueBootstrap}(c_i, TV, BK)$
4. for $j \leftarrow 1$ to $B^{d-i-2}$ do
5.    $TV_{j-1} \leftarrow \text{PublicKeySwitch}((\bar{c}_{(j-1)X_B}, ..., \bar{c}_{jX_B}), f, KS)$
6. return $\bar{c}_0$
Building Blocks Optimizations

- Base-aware key switching
  - A specialized key switching to pack $B < N$ samples.
  - $\frac{N}{B}$ times speedup over the generic key switching.

- Multi-value extract
  - Enables ciphertext scalings with linear error growth (instead of quadratic).
  - Improves error output variance of the multi-value bootstrap from quadratic to linear.
Base-Aware Key Switching

- Homomorphically calculate the phase $\phi(c) = b - \langle a, s \rangle$.
- For packing LWE samples:

$$
(0, f(b^{(1)}, b^{(2)}, \ldots, b^{(p)})) - \sum_{i=1}^{n} f(a^{(1)}_{i}, a^{(2)}_{i}, \ldots, a^{(p)}_{i}) \cdot KS_{i,j}
$$

- $f: \mathbb{T}^B \mapsto \mathbb{T}_N[X] = (a_1, \ldots, a_B) \mapsto a_1X^{N-1} + \ldots + a_B$
- $KS_{i,j}$ is a bit by bit TRLWE encryption of the key.
Base-Aware Key Switching

- Homomorphically calculate the phase $\phi(c) = b - \langle a, s \rangle$.
- For packing LWE samples:

$$
(0, f(b^{(1)}, b^{(2)}, \ldots, b^{(p)})) - \sum_{i=1}^{n} f(a^{(1)}_i, a^{(2)}_i, \ldots, a^{(p)}_i) \cdot KS_{i,j}
$$

- $f : \mathbb{T}^B \mapsto \mathbb{T}_N[X] = (a_1, \ldots, a_B) \mapsto a_1X^{N-1} + \ldots + a_B$
- $KS_{i,j}$ is a bit by bit TRLWE encryption of the key.

- Base Aware logic:
  - We want to pack $B < N$ samples.
  - $KS_{i,j,b} \in \text{TRLWE}_{s_{\text{base}}}(\frac{s_i}{\text{base}}, \sum_{q=bN/B}^{(b+1)N/B-1} X^q)$
  - $\frac{N}{B}$ times improvement in performance and error growth.
\[ \sigma_{x+y}^2 = \sigma_x^2 + \sigma_y^2 + 2\rho\sigma_x\sigma_y \]

- Addition of independent variables: \( \sigma_{x+y}^2 = \sigma_x^2 + \sigma_y^2 \)
- Multiplication by \( n \in \mathbb{Z} \): \( \sigma_{n\times x}^2 = n^2 \times \sigma_x^2 \)
- Can we implement multiplications as sequences of additions of independent TLWE samples?
LUT Encoding:

\[ L = [l_1, l_2, l_3, l_4] \mapsto \sum_{i=0}^{255} \frac{l_1}{8} X^i + \sum_{i=256}^{511} \frac{l_2}{8} X^i + \sum_{i=512}^{767} \frac{l_3}{8} X^i + \sum_{i=768}^{1023} \frac{l_4}{8} X^i \]

**Definition (Independence Heuristic\(^{14}\))**

The error of the coefficients of TRLWE samples (including TRGSW samples) and all linear combinations of them considered in TFHE are independent and concentrated.

---

\(^{14}\)Chillotti et al., “TFHE: fast fully homomorphic encryption over the torus”, 2020.
**Algorithm 1:** Multiplication (scaling) using the multi-value extract

**Input:** a TRLWE sample \( c \in \text{TRLWE}_S(p) \), which is the accumulator (ACC) of a previous functional bootstrap, and a cleartext scalar \( b \in \mathbb{Z} \).

**Output:** a TLWE sample \( \overline{c} \in \text{TLWE}_\overline{S}(b \cdot p_0) \), where \( p_0 \) is the constant term of \( p \), and \( \overline{S} \in \mathbb{B}^N \) is a vector interpretation of \( S \in \mathbb{B}_N[X] \).

1. \( \overline{c} \leftarrow \text{TLWE}_\overline{S}(0) \)
2. \( \overline{c} \leftarrow \overline{c} + \text{SampleExtract}_i(p) \), for each \( i \in \left[ 0, \left\lceil \frac{b}{2} \right\rceil - 1 \right] \)
3. \( \overline{c} \leftarrow \overline{c} - \text{SampleExtract}_i(p) \), for each \( i \in \left[ N - \left\lfloor \frac{b}{2} \right\rfloor, N - 1 \right] \)
4. Return \( \overline{c} \)
Multi-value Extract IV

Figure 2: Comparison between the variance of scaling using the multi-value extract and direct multiplication.
The multi-value extract allows evaluating any scalings with linear growth in the error variance.

In the Multi-value bootstrap of Carpov et al.\textsuperscript{15}:

- Direct multiplication: $\| TV_f \|^2 \leq s(q - 1)^2$
- Multi-value extract scaling: $\| TV_f \|^2 \leq s(q - 1)$

\textsuperscript{15}Sergiu Carpov, Malika Izabachène, and Victor Mollimard. “New Techniques for Multi-value Input Homomorphic Evaluation and Applications”. In: Topics in Cryptology – CT-RSA 2019. Ed. by Mitsuru Matsui. Cham: Springer International Publishing, 2019, pp. 106–126. ISBN: 978-3-030-12612-4. DOI: 10.1007/978-3-030-12612-4\_6.
Results
Results

• Implementations using the Functional Bootstrap
  • Generic LUT (6-bit-to-6-bit): 3.19x
  • Integer comparison (32-bit): 2.49x

• Implementations using Logic gates
  • ReLU (8-bit): 6.98x
  • Addition (8-bit): 8.74x
  • Maximum (8-bit): 3.5x
## 6-bit-to-6-bit LUT

|                     | Security | Key Size | Error Rate ($\log_2$) | Time (ms) | Speedup |
|---------------------|----------|----------|------------------------|-----------|---------|
| Carpov et al.\textsuperscript{16} | $\geq 128$ | $\approx 8$ GB | -26.94 | 1570\textsuperscript{a} | 1.00 |
| **This work (1)** | 127 | $\approx 4.3$ GB | -59.59 | 378.2 | 2.49 |
| **This work (2)** | 127 | $\approx 6.5$ GB | -134.84 | 457.9 | 2.06 |

\textsuperscript{a} Result provided by the authors, who executed experiments on a machine 1.67 times slower than ours. The speedup was adjusted accordingly.

\textsuperscript{16} Carpov, Izabachène, and Mollimard, “New Techniques for Multi-value Input Homomorphic Evaluation and Applications”, 2019.
## 32-bit Integer Comparison

|                | Security | Key Size | Error Rate (\(\log_2\)) | Time (ms) | Speedup |
|----------------|----------|----------|--------------------------|-----------|---------|
| Bourse *et al.*\(^{17}\) |          |          |                          |           |         |
| 90             | ≈ 1.2 GB | -50\(^b\) | 2232\(^a\)               | 1.75      |
| 109            | ≈ 3.4 GB | -47\(^b\) | 3902\(^a\)               | 1.00      |
| 211            | ≈ 4.6 GB | -89\(^b\) | 3840\(^a\)               | 1.02      |
| Zhou *et al.*\(^{18}\) |          |          |                          |           |         |
| 80             | ≈ 0.3 GB | Negligible | 1143.2                   | 0.93      |
| 127            | ≈ 0.3 GB | Negligible | 1867.2                   | 0.57      |
| **This work (1)** |         |          |                          |           |         |
| 127            | ≈ 4.3 GB | -26.51    | 334.1                    | 3.19      |
| **This work (2)** |         |          |                          |           |         |
| 127            | ≈ 6.5 GB | -129.58   | 396.4                    | 2.68      |

\(^a\) Execution time provided by the authors, who executed experiments on a machine 3.67 times slower than ours. The speedup was adjusted accordingly.

\(^b\) Error Rate provided by the authors. We speculatively estimate it to be much lower, but we do not have sufficient data to calculate.

\(^{17}\) Bourse, Sanders, and Traoré, “Improved Secure Integer Comparison via Homomorphic Encryption”, 2020.

\(^{18}\) Zhou et al., “Deep Binarized Convolutional Neural Network Inferences over Encrypted Data”, 2020.
## 8-bit Addition

|             | Security | Key Size | Error Rate \((\log_2)\) | Time (ms) | Speedup |
|-------------|----------|----------|--------------------------|-----------|---------|
| Lou and Jiang\(^{19}\) | 80       | \(\approx 0.3\) GB | Negligible               | 585       | 1.21    |
|             | 127      | \(\approx 0.3\) GB | Negligible               | 708.9     | 1.00    |
| Zhou et al.\(^{20}\)   | 80       | \(\approx 0.3\) GB | Negligible               | 338       | 2.10    |
|             | 127      | \(\approx 0.3\) GB | Negligible               | 548.7     | 1.29    |
| **This work (1)** | 127      | \(\approx 4.3\) GB | -124.7                   | 81.1      | 8.74    |
| **This work (2)** | 127      | \(\approx 6.5\) GB | -176.139                 | 94.8      | 7.48    |

---

\(^{19}\) Lou and Jiang, “SHE: A Fast and Accurate Deep Neural Network for Encrypted Data”, 2019.

\(^{20}\) Zhou et al., “Deep Binarized Convolutional Neural Network Inferences over Encrypted Data”, 2020.
Estimation of a full inference

**Table 3:** Estimation of an inference on the Binarized CNN of Zhou *et al.*\(^{21}\).

| Security Level | Execution time per layer (h) | Total (h) | Speedup |
|----------------|-------------------------------|-----------|---------|
|                | Bin. Conv. | Max-Pool. | Fully Conn. |       |
| Zhou *et al.* (reported) | 80 | 19.20 | 0.67 | 21.35 | 41.22 | 1.88 |
| Zhou *et al.* | 80 | 20.18 | 0.96 | 26.87 | 48.01 | 1.62 |
|                | 127 | 32.46 | 1.56 | 43.62 | 77.64 | 1.00 |
| This work (1) | 127 | 7.39 | 0.53 | 7.91 | 15.83 | 4.90 |
| This work (2) | 127 | 8.19 | 0.61 | 9.20 | 18.00 | 4.31 |

\(^{21}\)Zhou *et al.*, “Deep Binarized Convolutional Neural Network Inferences over Encrypted Data”, 2020.
Conclusion

- Two methods for combining Multiple Functional Bootstraps
  - Gains of up to 8 times over previous literature
  - The possibility of efficiently implementing functions with high precision.

- Building blocks optimizations
  - Specialized packing key switching: 256 times performance and error improvements over the generic technique.
  - Multi-value extract: scaling with linear error growth.

- Complete error analysis with experimental validation.
Thank you!\textsuperscript{22}

\textsuperscript{22}We would like to thank the São Paulo Research Foundation (grants 2013/08293-7 and 2019/12783-6), and the National Council for Scientific and Technological Development (grant 313012/2017-2) for supporting this research. We also would like to thank Tancrède Lepoint for serving as the shepherd for this paper.