Amortized Functional Bootstrapping for Homomorphic Evaluation of Encrypted Functions

Yan Xu[1,2,3], Li-Ping Wang[1,2,3(✉)], and Huaxiong Wang[4]

[1] Key Laboratory of Cyberspace Security Defense, Institute of Information Engineering, CAS, Beijing, China
{xuyan1,wangliping}@iie.ac.cn
[2] State Key Laboratory of Cryptology, Beijing, China
[3] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[4] School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore
hxwang@ntu.edu.sg

Abstract. Functional bootstrapping is for a cloud server to refresh the error of a ciphertext and homomorphically evaluate a public function f without decryption. Furthermore, in some applications, a cloud server needs to perform functional bootstrapping without knowing f but given a certain encryption of f. Currently, the best amortized asymptotic complexity of the amortized variants of functional bootstrapping is $\tilde{O}(1)$ per message for a public function. However, there is no amortized algorithm for an encrypted function. In this paper, we propose a novel amortized functional bootstrapping algorithm whether the function f is public or encrypted. Firstly, we extend the amortized homomorphic automorphism technique of LW23 (Liu and Wang, EUROCRYPT 2023) to perform different automorphisms and key switching on multiple ciphertexts simultaneously. Then, by using our extended automorphism technique, we improve the homomorphic inverse number theoretic transform (NTT) technique of Guimarães et al. (ASIACRYPT 2023) to reduce costs. Next, we combine the amortized bootstrapping framework of Micciancio and Sorrell (ICALP 2018) with the above improved inverse NTT technique to construct our amortized functional bootstrapping algorithm whose amortized cost is $\tilde{O}(1)$. Moreover, the sub-Gaussian parameters of the output errors in our algorithm are $\tilde{O}(EN^{9.375})$ where N is the number of input ciphertexts and E is the sub-Gaussian parameter of the errors of bootstrapping keys, and thus they are independent of f. In particular, they are reduced by a factor of $\tilde{O}(|f(x)|N^{28.625})$ for any public function f, compared with the work of Liu and Wang (ASIACRYPT 2023).

Keywords: Fully homomorphic encryption · Functional bootstrapping · FHEW · LWE.

1 Introduction

Fully homomorphic encryption (FHE) allows arbitrary computation over encrypted data without decryption and is widely used in secure outsourcing computing, privacy-preserving machine learning, private information retrieval. In FHE, noisy ciphertexts are used to ensure security, and the noise causes an error in the ciphertext. The error grows with homomorphic operations and may become too large to retrieve the plaintexts. Gentry [12] introduced bootstrapping to refresh a ciphertext and reduce its error, and he constructed the first plausible FHE scheme using bootstrapping.

Gentry's original bootstrapping was extremely impractical, and improving its effectiveness has been the main goal of many papers to make FHE faster. Most of these works are based either on learning with errors (LWE) [29] or on its ring version RLWE [23]. At present, the most efficient bootstrapping algorithms, such as [10,7], are constructed by the Gentry-Sahai-Waters (GSW) scheme [13,1]. The FHEW scheme [10] extends the GSW scheme to a ring version RGSW and uses it to reduce the bootstrapping time to within 1 second, and the work of [7] introduced the RGSW-RLWE external product to improve the bootstrapping time to only 13 ms. Following the framework of bootstrapping in [10,7], subsequent works further constructed more useful functional bootstrapping algorithms, also known as programmable bootstrapping algorithms, such as [2,8,6,9,14,17,21,25]. Given a function f and a ciphertext $\boldsymbol{c}$ of a plaintext μ, the functional bootstrapping refreshes the noise of $\boldsymbol{c}$ and evaluates $f(\mu)$ homomorphically. Furthermore, when f is encrypted, the algorithms in [2,8,9] can also calculate $f(\mu)$ homomorphically. But all of these works can only process one plaintext at a time.

To batch the functional bootstrapping, Micciancio and Sorrell [26] packed N LWE ciphertexts into an RLWE ciphertext using the public functional key switching of [7]. Then, based on the Nussbaumer transform and their slow multiplication technique, they homomorphically implemented the decryption of the RLWE ciphertext on an exponent using the FHEW scheme, thereby reducing the amortized cost per message from $O(N)$ to $O(3^{\rho}N^{\frac{1}{\rho}})$ for any $\rho > 0$. In [15,27], the homomorphic automorphism of [18] is used to improve the homomorphic exponentiation operations in the bootstrapping of [26], and the Nussbaumer transform is replaced with the standard number theoretic transform (NTT) to make bootstrapping more straightforward and practical, further reducing the amortized cost to $O(\rho N^{\frac{1}{\rho}})$. Liu and Wang [19,20] proposed an amortized version of the homomorphic multiplication of [10,7] based on the properties of tensor rings and trace functions. Additionally, they applied this technique to reduce the amortized cost of the algorithm of [26] to $\tilde{O}(1)$ in [20]. The amortized cost of the amortized functional bootstrapping algorithm of [22] is also $\tilde{O}(1)$, but the idea of [22] does not follow the line of [26,15,27,19,20]. The algorithm of [22] uses the single instruction multiple data (SIMD) property of the Brakerski-Fan-Vercauteren (BFV) scheme [4,3,11] and involves homomorphic encoding (i.e., packing and decryption), turning function evaluation into polynomial computation, homomorphic decoding, and extraction. In addition, multi-bit plaintexts are supported in [15,22], whereas only single-bit plaintexts are supported in

[26,27,19,20]. However, in [15,22], the output errors are related to f, and only public functions can be evaluated.

Moreover, some applications need homomorphic evaluations of encrypted functions. Think about the following application scenario: To save local memory consumption, a client has encrypted many private messages $\{\mu_0, \mu_1, \dots\}$ into ciphertexts, stored the ciphertexts on a cloud server, and then deleted the messages locally. Now, the client wants to evaluate a valuable function f, such as a machine learning model developed with enormous resources, on these messages while preserving the confidentiality of f and the messages. One straightforward method is for the client to get all of the ciphertexts from the cloud server, decrypt them to obtain $\{\mu_0, \mu_1, \dots\}$, evaluate f on the messages, encrypt the results $\{f(\mu_0), f(\mu_1), \dots\}$ into new ciphertexts, and then send the new ciphertexts to the cloud server. There are substantial costs associated with the client's local storage, the client's local computing, and the communication between the client and the cloud server in this method. However, if an algorithm enables the homomorphic evaluation of encrypted functions, there is a much easier and cheaper way. The client only needs to encrypt f and send the ciphertext of f to the cloud server. Then the cloud server evaluates $\{f(\mu_0), f(\mu_1), \dots\}$ homomorphically using the ciphertexts of $\{\mu_0, \mu_1, \dots\}$ and f by this algorithm.

Therefore, in this work, we explore how to construct a functional bootstrapping algorithm whose amortized complexity is $\tilde{O}(1)$ and whose output errors are independent of f to evaluate any function f on multi-bit plaintexts, whether the function f is public or encrypted.

Contributions. Firstly, using the packing method of [19,20], we extend the amortized homomorphic automorphism algorithm of [20]. The algorithm of [20] only allows applying the same automorphism to multiple ciphertexts each time. However, our improved version can perform various automorphisms and key switching on these ciphertexts simultaneously.

Then, based on the amortized RGSW-RLWE multiplication of [19] and our extended automorphism algorithm, we improve the homomorphic inverse NTT algorithm of [15] to reduce costs. Our inverse NTT allows the plaintexts of the input and output ciphertexts to be in the form of vX^{φ}, which is a crucial property for evaluating an encrypted function f, whereas the form is X^{φ} in prior works, where v is a test polynomial encoding f and φ is an integer.

Next, following the amortized bootstrapping framework of [26], using our improved inverse NTT algorithm, we propose a new amortized functional bootstrapping algorithm whose amortized complexity is $\tilde{O}(1)$ to compute any public or encrypted function f on multi-bit plaintexts. Moreover, the sub-Gaussian parameters of the output errors are $\tilde{O}(EN^{9.375})$ and thus are independent of f, where N is the number of the input ciphertexts and E is the sub-Gaussian parameter of the errors of bootstrapping keys. In particular, for any public function f, the sub-Gaussian parameters of the output errors are reduced by at least a factor of $\tilde{O}(|f(x)|N^{28.625})$ compared with [22]. Below, we further introduce the idea of our amortized functional bootstrapping algorithm.

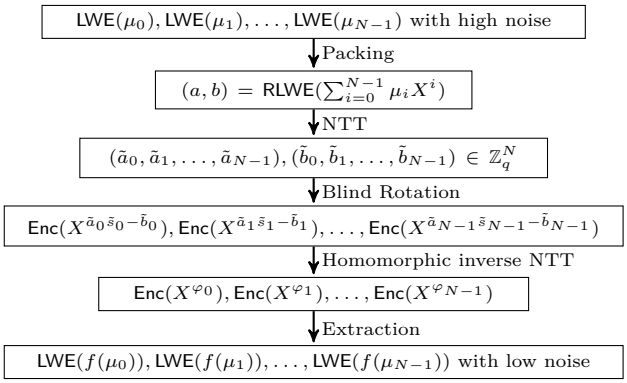

Fig. 1. The framework of amortized FHEW-like functional bootstrapping

Technical Overview. We first give a quick review of the framework of the amortized FHEW-like functional bootstrapping of [26] and then present the technical details of our amortized functional bootstrapping algorithm.

Similar to [26,15,27,20], our amortized functional bootstrapping follows the following framework. Consider inputting N high-noise LWE ciphertexts corresponding to plaintexts $\mu_0, \mu_1, \ldots, \mu_{N-1}$ together with a function f. Firstly, employing the packing technique of [26], pack the ciphertexts into an RLWE ciphertext (a, b) such that $b - a \cdot s = \sum_{i=0}^{N-1} \mu_i X^i + e \pmod q$ where s is its secret key and $e = \sum_{i=0}^{N-1} e_i X^i$ is its error. Then, run NTTs on a and b, obtaining integer vectors $(\tilde{a}_0, \ldots, \tilde{a}_{N-1})$ and $(\tilde{b}_0, \ldots, \tilde{b}_{N-1})$, respectively. Next, perform blind rotation on a ciphertext of $X^{\tilde{s}_i}$ to produce $\mathsf{Enc}(X^{\tilde{\varphi}_i})$, which is a ciphertext of $X^{\tilde{\varphi}_i}$ and is kept in the register of the bootstrapping procedure, where $(\tilde{s}_0, \ldots, \tilde{s}_{N-1}) = \mathsf{NTT}(s)$ and $\tilde{\varphi}_i = \tilde{a}_i \cdot \tilde{s}_i - \tilde{b}_i$. After that, perform the inverse NTT on the exponents of $\{X^{\tilde{\varphi}_i}\}$ homomorphically to get ciphertexts $\{\mathsf{Enc}(X^{\varphi_i})\}$, where $-\varphi_i = \mu_i + e_i \pmod q$. Finally, extract a low-noise LWE ciphertext of $f(\mu_i)$ from $\mathsf{Enc}(X^{\varphi_i})$. Figure 1 depicts the development of the input ciphertexts in this framework.

Based on the framework and the advantages of the previous related works, we construct a novel amortized functional bootstrapping algorithm. For the NTT, we adopt the standard NTT as in [15,27] to make bootstrapping easier, instead of using the Nussbaumer transform like in [26,20].

Additionally, blind rotation and homomorphic inverse NTT mainly include homomorphic multiplications and homomorphic exponentiations. For homomorphic multiplications, we use the amortized RGSW-RLWE external products of [19] for best efficiency. Besides, for homomorphic exponentiations, the current best method is using the amortized homomorphic automorphism algorithm of [20]. However, this algorithm can only perform the same homomorphic exponentiation operation on multiple inputs each time. Therefore, we extend the automorphism algorithm of [20] by utilizing larger tensor rings and the batching technique of [19,20] to perform different homomorphic exponentiation opera-

tions on multiple inputs simultaneously. Moreover, we improve the homomorphic inverse NTT algorithm of [15] by employing the amortized RGSW-RLWE multiplication of [19] and our extended amortized homomorphic automorphism algorithm to reduce computational costs and make the outputs suitable for our bootstrapping. Particularly, the design of our improved inverse NTT algorithm carefully considers the impact of a test polynomial v encoding the evaluated function f throughout the inverse NTT process, while inverse NTT algorithms of [26,15,27,20] have not considered v.

Furthermore, for extraction, the bootstrapping algorithms of [26,27,19,20] directly use the extraction procedure of [10,7], but they only evaluate a public Boolean function f. For any public function f, the bootstrapping algorithm of [15] multiplies the results of the inverse NTT by the test polynomial v encoding f homomorphically and then executes the extraction of [7], producing outputs whose errors are related to f. However, our bootstrapping algorithm multiplies v before blind rotation and uses the extraction of [27], making the output errors independent of f, as stated in [6]. This also makes our algorithm slightly different from the framework in Figure 1 used in [26,15,27,20]. In our amortized functional bootstrapping algorithm, the blind rotation operation outputs $\{\mathsf{Enc}(vX^{\tilde{a}_0\tilde{s}_0-\tilde{b}_0}), \mathsf{Enc}(X^{\tilde{a}_1\tilde{s}_1-\tilde{b}_1}), \ldots, \mathsf{Enc}(X^{\tilde{a}_{N-1}\tilde{s}_{N-1}-\tilde{b}_{N-1}})\}$, and the inverse NTT operation outputs $\{\mathsf{Enc}(vX^{\varphi_0}), \mathsf{Enc}(vX^{\varphi_1}), \ldots, \mathsf{Enc}(vX^{\varphi_{N-1}})\}$.

Organization. The rest of the paper is organized as follows. In Section 2, we describe some necessary background on functional bootstrapping and packing based on tensor rings. In Section 3, we introduce our extended amortized homomorphic automorphism algorithm, our improved homomorphic inverse NTT algorithm, and our novel amortized functional bootstrapping algorithm. In Section 4, we conclude the paper.

2 Preliminaries

Notations. The set of integers is denoted by $\mathbb{Z}$ and its quotient ring is denoted by $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ with $\left[-\frac{q}{2}, \frac{q}{2}\right)$ as the representative interval. The set of positive integers is denoted by $\mathbb{Z}^+$. For $a \in \mathbb{Z}$, define $[a]_q = a \bmod q \in [-\frac{q}{2}, \frac{q}{2})$ and $[a]_q^+ = a \bmod^+ q \in [0, q)$. Column vectors are denoted by bold lowercase letters, e.g. $\boldsymbol{a}$, and matrices are denoted by bold capital letters, e.g. $\boldsymbol{A}$. Define $[n] = \{0, 1, \ldots, n-1\}$ and define $\langle \boldsymbol{a}, \boldsymbol{b} \rangle$ as the inner product of $\boldsymbol{a}$ and $\boldsymbol{b}$. We also use $(a_0, \ldots, a_{n-1})$ or $(a_i)_{i\in[n]}$ to represent a column vector. If D is a set, then $a \xleftarrow{\$} D$ denotes sampling a uniformly from D. If χ is a probability distribution, then $a \xleftarrow{\$} \chi$ denotes sampling a according to χ. For a polynomial $a = \sum_{i=0}^{N-1} a_i X^i$, denote its coefficient vector by $\boldsymbol{a} = (a_0, \ldots, a_{N-1})$ and define its norms as $\|a\| = \|\boldsymbol{a}\| = \max_{i\in[N]} |a_i|$, $\|a\|_1 = \|\boldsymbol{a}\|_1 = \sum_{i=0}^{n-1} |a_i|$ and $\|a\|_2 = \|\boldsymbol{a}\|_2 = \sqrt{\sum_{i=0}^{n-1} a_i^2}$. Besides, $\lfloor . \rceil$, $\lfloor . \rfloor$, and $\lceil . \rceil$ denote the rounding, floor, and ceiling functions, respectively. For a ring $\mathcal{R}$, define $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$ and define the expansion factor of $\mathcal{R}$ as $\gamma_{\mathcal{R}} = \max\{\frac{\|a\cdot b\|}{\|a\|\cdot\|b\|} : a, b \in \mathcal{R}\}$. All logarithms are base 2 unless otherwise noted.

2.1 Basic Tools of Bootstrapping

Our functional bootstrapping algorithm includes packing LWE ciphertexts, blind rotation, homomorphic inverse NTT, and extraction. Specifically, blind rotation and homomorphic inverse NTT involve external products between RGSW ciphertexts and RLWE ciphertexts, as well as homomorphic automorphisms. Besides, we store register (REG) ciphertexts in the registers of our algorithm, and RGSW ciphertexts can be obtained by scheme switching from REG ciphertexts. The introduction of these tools is presented as follows.

(R)LWE. Our algorithms are based on learning with errors (LWE) and its ring version RLWE introduced by [29,23]. We review the (R)LWE problem and define (R)LWE ciphertexts in Definition 1. And there is strong evidence showing the hardness of the LWE problem, e.g. [5,29], and the RLWE problem, e.g. [23,24,28].

Definition 1. *Let $q, n, \Delta \in \mathbb{Z}^+$. Let χ be a distribution over $\mathbb{Z}$, and let $\mathsf{sk} \in \mathbb{Z}_q^n$ be the secret key. A sample from the LWE distribution $A_{\mathsf{sk},\chi}$ is of the form $(\boldsymbol{a}, b) \in \mathbb{Z}_q^{n+1}$ with $b = [\langle \boldsymbol{a}, \mathsf{sk} \rangle + e]_q$, where $\boldsymbol{a} \xleftarrow{\$} \mathbb{Z}_q^n$ and $e \xleftarrow{\$} \chi$. The decision LWE problem is to distinguish the uniform distribution over $\mathbb{Z}_q^{n+1}$ and $A_{\mathsf{sk},\chi}$. Define an LWE ciphertext of $\mu \in \mathbb{Z}$ as $\mathsf{ct} = (\boldsymbol{a}, [b+\Delta\mu]_q) \in \mathbb{Z}_q^{n+1}$. Use $\mathsf{LWE}_{\mathsf{sk}}^{q,\Delta}(\mu)$ to denote the set of all LWE ciphertexts of the plaintext μ under the secret key sk with the ciphertext modulus q and the scale factor Δ.*

For a cyclotomic ring $\mathcal{R}$, let χ be a distribution over $\mathcal{R}$, and let $\mathsf{sk} \in \mathcal{R}_q$ be the secret key. A sample from the RLWE distribution $A_{\mathsf{sk},\chi}$ is of the form $(a, b) \in \mathcal{R}_q^2$ with $b = [a \cdot \mathsf{sk} + e]_q$, where $a \xleftarrow{\$} \mathcal{R}_q$ and $e \xleftarrow{\$} \chi$. The decision RLWE problem is to distinguish the uniform distribution over $\mathcal{R}_q^2$ and $A_{\mathsf{sk},\chi}$. Define an RLWE ciphertext of $\mu \in \mathcal{R}$ as $\mathsf{ct} = (a, [b + \Delta\mu]_q) \in \mathcal{R}_q^2$. Use $\mathsf{RLWE}_{\mathsf{sk},\mathcal{R}}^{q,\Delta}(\mu)$ to denote the set of all RLWE ciphertexts of the plaintext μ under the secret key sk with the ciphertext modulus q and the scale factor Δ over the ring $\mathcal{R}$.

Sub-Gaussian. As discussed in [1,10], it is convenient to use the notion of sub-Gaussian to analyze the error growth in the FHE constructions. A variable x is sub-Gaussian with parameter $\sigma > 0$ if $\mathbb{E}[\exp(2\pi kx)] \leq \exp(\pi\sigma^2 k^2)$ for any real number k. Then a sub-Gaussian variable x with parameter $\sigma > 0$ satisfies $\mathsf{Pr}[|x| \geq k] \leq 2\exp(-\pi k^2/\sigma^2)$ for a real number k. Additionally, if x_i is a sub-Guassian variable with parameter σ_i for $i \in [n]$ and $\{x_i\}_{i\in[n]}$ are independent, then for any real numbers $\{a_0, \ldots, a_{n-1}\}$, $\sum_{i=0}^{n-1} a_i x_i$ is sub-Guassian with parameter $\sqrt{\sum_{i=0}^{n-1} a_i^2\sigma_i^2}$. Moreover, a polynomial (or a vector) is sub-Guassian with parameter σ if its coefficients are independent and sub-Guassian with parameter σ_i where $\sigma_i \leq \sigma$. Besides, it follows directly from the definition that the concatenation of variables or vectors, each of which is sub-Guassian with parameter σ, is also sub-Guassian with parameter σ. To derive average bounds for the errors of the ciphertexts, we use the well-known and commonly used independence heuristic: All the coefficients of the error terms of the LWE and

RLWE samples appearing in the linear combinations we consider are assumed to be independent and concentrated. More precisely, they are sub-Guassian.

In addition, the gadget vector is defined as $\boldsymbol{g} = (1, B, \ldots, B^{\ell-1}) \in \mathbb{Z}^{\ell}$ where $\ell = \lceil \log_B Q \rceil$ and $B, Q \in \mathbb{Z}^+$. As claimed in [1], for any $a \in \mathbb{Z}_Q$, there is a randomized and efficiently computable function $\mathsf{g}^{-1} : \mathbb{Z}_Q \to \mathbb{Z}^{\ell}$ such that $\boldsymbol{x} = \mathsf{g}^{-1}(a)$ is sub-Gaussian with parameter $O(1)$ and always satisfies $\langle \boldsymbol{g}, \boldsymbol{x} \rangle = a$. The gadget matrix is defined as $\boldsymbol{G} = \boldsymbol{I}_2 \otimes \boldsymbol{g} \in \mathbb{Z}^{2\ell \times 2}$, where $\otimes$ denotes the Kronecker product. And g^{-1} is extended to the vector (or matrix) case by applying g^{-1} to each entry of the vector (or matrix), using the notation G^{-1}. And g^{-1} is extended to the polynomial case by applying g^{-1} to each coefficient of the polynomial.

REG and RGSW. Below, we define the REG ciphertexts stored in our registers as the gadget RLWE ciphertexts of [27], use the REG ciphertexts to represent the RGSW ciphertexts of [10,7], and introduce related homomorphic operations.

Definition 2. *Let $Q, B, \Delta \in \mathbb{Z}^+$ and $\ell = \lceil \log_B Q \rceil$. For a cyclotomic ring $\mathcal{R}$, let $\mathsf{sk} \in \mathcal{R}$ be the secret key. Define an REG ciphertext of $\mu \in \mathcal{R}$ as $\boldsymbol{C}_{\mathsf{reg}} = (\boldsymbol{c}_0^{\top}, \boldsymbol{c}_1^{\top}, \ldots, \boldsymbol{c}_{\ell-1}^{\top}) \in \mathcal{R}_Q^{\ell \times 2}$ where $\boldsymbol{c}_i \in \mathsf{RLWE}_{\mathsf{sk},\mathcal{R}}^{Q,B^i}(\mu)$ for $i \in [\ell]$. Use $\mathsf{REG}_{\mathsf{sk},\mathcal{R}}^{Q}(\mu)$ to denote the set of all REG ciphertexts of the plaintext μ under the secret key sk with the ciphertext modulus Q over the ring $\mathcal{R}$. Define an RGSW ciphertext of $\mu \in \mathcal{R}$ as $\boldsymbol{C}_{\mathsf{rgsw}} = (\boldsymbol{C}_{\mathsf{reg},0}, \boldsymbol{C}_{\mathsf{reg},1}) \in \mathcal{R}_Q^{2\ell \times 2}$ where $\boldsymbol{C}_{\mathsf{reg},0} \in \mathsf{REG}_{\mathsf{sk},\mathcal{R}}^{Q}(-\mathsf{sk} \cdot \mu)$ and $\boldsymbol{C}_{\mathsf{reg},1} \in \mathsf{REG}_{\mathsf{sk},\mathcal{R}}^{Q}(\mu)$. Use $\mathsf{RGSW}_{\mathsf{sk},\mathcal{R}}^{Q}(\mu)$ to denote the set of all RGSW ciphertexts of μ under the secret key sk with the ciphertext modulus Q over the ring $\mathcal{R}$. For homomorphic multiplications, given $d \in \mathcal{R}_Q$ and $\boldsymbol{c} \in \mathsf{RLWE}_{\mathsf{sk},\mathcal{R}}^{Q,\Delta}(\mu')$, define external products $\boldsymbol{C}_{\mathsf{reg}} \odot d = [\boldsymbol{C}_{\mathsf{reg}}^{\top} \mathsf{g}^{-1}(d)]_Q$ and $\boldsymbol{C}_{\mathsf{rgsw}} \boxtimes \boldsymbol{c} = [\boldsymbol{C}_{\mathsf{rgsw}}^{\top} \mathsf{G}^{-1}(\boldsymbol{c})]_Q$.*

From here, we denote B as a gadget base, Δ as a scale factor, N as the number of the input ciphertexts of bootstrapping, and $\mathcal{R}$ as a cyclotomic ring. Denote n and q as the dimension and modulus of the input ciphertexts of bootstrapping, respectively. Denote Q as the modulus of bootstrapping keys. Set $B, \Delta, Q \in \mathbb{Z}^+$ and $\ell = \lceil \log_B Q \rceil$. Set N as a power of two, and set q as an odd prime number satisfying $q = 1 \pmod{2N}$. Denote z as the secret key of bootstrapping, and set $z = \sum_{i=0}^{q-2} z_i X^i$ and $\boldsymbol{z}' = (-z, 1)$ where $z_i \in \{0, 1, -1\}$ for $i \in [q-1]$. The correctness of the external products defined above is shown in Lemma 1.

Lemma 1 ([27]). *Let $\boldsymbol{C}_0 \in \mathsf{REG}_{z,\mathcal{R}}^{Q}(\mu_0)$ and $\boldsymbol{C}_1 \in \mathsf{RGSW}_{z,\mathcal{R}}^{Q}(\mu_1)$. Let $\boldsymbol{e}_0$ and $\boldsymbol{e}_1$ be the errors of $\boldsymbol{C}_0$ and $\boldsymbol{C}_1$, respectively, and be sub-Gaussian with parameter E. Then $\boldsymbol{C}_0 \boldsymbol{z}' = \mu_0 \boldsymbol{g} + \boldsymbol{e}_0 \pmod{Q}$ and $\boldsymbol{C}_1 \boldsymbol{z}' = \mu_1 \boldsymbol{G} \boldsymbol{z}' + \boldsymbol{e}_1 \pmod{Q}$. Let $d \in \mathcal{R}_Q$ and $\boldsymbol{c} \in \mathsf{RLWE}_{z,\mathcal{R}}^{Q,\Delta}(\mu)$ with an error e. Then $\boldsymbol{C}_0 \odot d \in \mathsf{RLWE}_{z,\mathcal{R}}^{Q,1}(d\mu_0)$ with an error $e_0' = \boldsymbol{e}_0^{\top} \mathsf{g}^{-1}(d)$, and $\boldsymbol{C}_1 \boxtimes \boldsymbol{c} \in \mathsf{RLWE}_{z,\mathcal{R}}^{Q,\Delta}(\mu\mu_1)$ with an error $e_1' = \mu_1 e + \hat{e}_1'$, where $\hat{e}_1' = \boldsymbol{e}_1^{\top} \mathsf{G}^{-1}(\boldsymbol{c})$, e_0' and $\hat{e}_1'$ are sub-Gaussian with parameter $O(E\sqrt{\gamma_{\mathcal{R}} \ell})$.*

Note that we can extend any homomorphic operation defined on an RLWE ciphertext to an REG ciphertext by applying this operation to each RLWE ciphertext in the REG ciphertext.

Packing LWE Ciphertexts. Given multiple LWE ciphertexts under the secret key $\boldsymbol{s}_{\mathsf{in}} = (s_{\mathsf{in},j})_{j\in[n]} \in \mathbb{Z}_q^n$, one can use the packing algorithm $\mathsf{PackLWE}$ of [15] to pack them into an RLWE ciphertext. Let $\boldsymbol{PaK}$ be the packing key where $\boldsymbol{PaK}_j \in \mathsf{REG}^q_{s,\hat{\mathcal{R}}}(s_{\mathsf{in},j})$ for $j \in [n]$, $s \in \hat{\mathcal{R}}_q$, $\hat{\mathcal{R}} = \mathbb{Z}[X]/(X^N+1)$. The following is a brief introduction to $\mathsf{PackLWE}$.

- $\mathsf{PackLWE}(\{\boldsymbol{c}_i\}_{i\in[N]}, \boldsymbol{PaK})$: Given ciphertexts $\{\boldsymbol{c}_i\}_{i\in[N]}$ and a packing key $\boldsymbol{PaK} = (\boldsymbol{PaK}_j)_{j\in[n]}$, where $\boldsymbol{c}_i = (a_{i,0}, \ldots, a_{i,n-1}, b_i) \in \mathsf{LWE}^{q,\Delta}_{\boldsymbol{s}_{\mathsf{in}}}(\mu_i)$ for $i \in [N]$, set $b' = \sum_{i=0}^{N-1} b_i X^i$ and $a'_j = \sum_{i=0}^{N-1} a_{i,j} X^i$ for $j \in [n]$. Compute $\boldsymbol{c}' = [(0,b') - \sum_{j=0}^{n-1} \boldsymbol{PaK}_j^\top \mathsf{g}^{-1}(a'_j)]_q$. Output $\boldsymbol{c}'$. Then $\boldsymbol{c}' \in \mathsf{RLWE}^{q,\Delta}_{s,\hat{\mathcal{R}}}(\sum_{i=0}^{N-1} \mu_i X^i)$.

Automorphism. Given a ciphertext of $\mu \in \mathcal{R}$, an automorphism θ of $\mathcal{R}$, one can use the homomorphic automorphism algorithm Aut of [27] to compute a ciphertext of $\theta(\mu)$. Let $\boldsymbol{AK}_\theta \in \mathsf{REG}^Q_{z,\mathcal{R}}(\theta(z))$ be the automorphism key with an error sub-Gaussian with parameter E. The process of Aut is shown below, and Lemma 2 demonstrates its correctness.

- $\mathsf{Aut}(\boldsymbol{c}, \theta, \boldsymbol{AK}_\theta)$: Given an RLWE ciphertext $\boldsymbol{c} = (a,b) \in \mathcal{R}_Q^2$, an automorphism θ of $\mathcal{R}$, and an automorphism key $\boldsymbol{AK}_\theta$, compute $\boldsymbol{c}_{\mathsf{aut}} \leftarrow (0, \theta(b)) - \boldsymbol{AK}_\theta \odot \theta(a)$ and then output $\boldsymbol{c}_{\mathsf{aut}} \in \mathcal{R}_Q^2$.

Lemma 2 ([27]). *Let $\boldsymbol{c} \in \mathsf{RLWE}^{Q,\Delta}_{z,\mathcal{R}}(\mu)$ with an error e such that $\boldsymbol{c}^\top \boldsymbol{z}' = \Delta\mu + e \pmod Q$ where $\boldsymbol{z}' = (-z, 1)$. If $\boldsymbol{c}_{\mathsf{aut}} \leftarrow \mathsf{Aut}(\boldsymbol{c}, \theta, \boldsymbol{AK}_\theta)$, then $\boldsymbol{c}_{\mathsf{aut}}^\top \boldsymbol{z}' = \theta(\Delta\mu + e) + e_{\mathsf{aut}} \pmod Q$ where e_{aut} is sub-Gaussian with parameter $O(E\sqrt{\gamma_{\mathcal{R}}\ell})$.*

Scheme Switching. One can apply the scheme switching of [27] to get an RGSW ciphertext from an REG ciphertext, maintaining the plaintext unchanged. Let $\boldsymbol{SwK} \in \mathsf{REG}^Q_{z,\mathcal{R}}(z^2)$ be the scheme switching key with an error sub-Gaussian with parameter E. We briefly introduce the scheme switching technique below.

- $\mathsf{Switch}(\boldsymbol{C}, \boldsymbol{SwK})$: Given an REG ciphertext $\boldsymbol{C} = (\boldsymbol{c}_0^\top, \ldots, \boldsymbol{c}_{\ell-1}^\top) \in \mathcal{R}_Q^{\ell\times 2}$ and a switching key $\boldsymbol{SwK}$, compute $\boldsymbol{c}'_i \leftarrow (c_{i,1}, 0) + \boldsymbol{SwK} \odot c_{i,0}$ where $\boldsymbol{c}_i = (c_{i,0}, c_{i,1})$ for $i \in [\ell]$, and then output $\boldsymbol{C}_{\mathsf{swi}} = ((\boldsymbol{c}'^\top_0, \ldots, \boldsymbol{c}'^\top_{\ell-1}), \boldsymbol{C}) \in \mathcal{R}_Q^{2\ell\times 2}$.

Lemma 3 ([27]). *Let $\boldsymbol{C} \in \mathsf{REG}^Q_{z,\mathcal{R}}(\mu)$ with an error $\boldsymbol{e}$. If we set $\boldsymbol{C}_{\mathsf{swi}} \leftarrow \mathsf{Switch}(\boldsymbol{C}, \boldsymbol{SwK})$. Then $\boldsymbol{C}_{\mathsf{swi}} \in \mathsf{RGSW}^Q_{z,\mathcal{R}}(\mu)$ with an error $\boldsymbol{e}_{\mathsf{swi}} = (-z\cdot\boldsymbol{e} + \boldsymbol{e}', \boldsymbol{e})$ where $\boldsymbol{e}'$ is sub-Gaussian with parameter $O(E\sqrt{\gamma_{\mathcal{R}}\ell})$.*

Extraction. Given an RLWE ciphertext of $vX^{-\varphi}$ over a ring $\mathcal{R}$, if $\varphi \in [q-1]$, $\mathcal{R} = \mathbb{Z}[X]/(\Phi_q(X))$ where $\Phi_q(X) = \sum_{i=0}^{q-1} X^i$, and $v = \sum_{i=0}^{q-2}\sum_{j=0}^{i} F_j X^i$ like in [16] where $F_j \in \mathbb{Z}$ for $j \in [q-1]$, then we can extract an LWE ciphertext of F_φ by using part of the steps in the $\mathsf{msbExtract}$ of [27]. The following is the extraction procedure, and the correctness is shown in Lemma 4.

– $\mathsf{Extract}(\boldsymbol{c}, Q)$ Given an RLWE ciphertext $\boldsymbol{c} = (\sum_{i=0}^{q-2} c_{0,i}X^i, \sum_{i=0}^{q-2} c_{1,i}X^i) \in \mathcal{R}_Q^2$ and its modulus Q, compute $c'_i \leftarrow [c_{0,[-i]_q} - c_{0,q-1-i}]_Q$ for $i = 0, \ldots, q-2$ and $c'_{q-1} \leftarrow c_{1,0}$, where $c_{0,q-1} = 0$. Output $\boldsymbol{c}' = (c'_0, \ldots, c'_{q-1}) \in \mathbb{Z}_Q^q$.

Lemma 4. *For a ring $\mathcal{R} = \mathbb{Z}[X]/(\Phi_q(X))$ where $\Phi_q(X) = \sum_{i=0}^{q-1} X^i$, let $v = \sum_{i=0}^{q-2}\sum_{j=0}^{i} F_j X^i \in \mathcal{R}$ where $F_j \in \mathbb{Z}$ for $j \in [q-1]$. Let $\boldsymbol{c} \in \mathsf{RLWE}_{z,\mathcal{R}}^{Q,1}(\mu)$ with an error e where $\mu = vX^{-\varphi}$, $\varphi \in [q-1]$ and $e = \sum_{i=0}^{q-2} e_i X^i$. Then, $\mathsf{Extract}(\boldsymbol{c}, Q)$ outputs an LWE ciphertext $\boldsymbol{c}' \in \mathbb{Z}_Q^q$ satisfying $\langle \boldsymbol{c}', (-\boldsymbol{z}, 1)\rangle = F_\varphi + e_0 \pmod Q$.*

Proof. We know that $\boldsymbol{c}^\top \boldsymbol{z}' = \sum_{i=0}^{q-2}(c_{1,i} - \sum_{j=0}^{q-2}(c_{0,[i-j]_q} - c_{0,q-1-j}) \cdot z_j)X^i = \mu + e \pmod Q$ where $\boldsymbol{z}' = (-z, 1)$. Based on Proposition 1 of [16], over the ring $\mathcal{R}$, the constant term of vX^{-k} is F_k for $k \in [q-1]$. So $\langle \boldsymbol{c}', (-\boldsymbol{z}, 1)\rangle = F_\varphi + e_0 \pmod Q$.

2.2 Amortized External Product

Our bootstrapping algorithm uses the techniques of [19,20] to optimize the external products in Definition 2. The optimization utilizes the properties of tensor rings to pack multiple ciphertexts and then employs trace functions to batch multiplications and unpack. In the following text, we first introduce additional notations about tensor rings and then describe the packing method, the homomorphic evaluation method of a trace function, the amortized external products and the unpacking method, all of which are introduced in detail in [19,20].

Notations. Let K, K_0, K_1 and K_2 be linearly disjoint fields, where $K = \mathbb{Q}[\xi_q]$, $K_i = \mathbb{Q}[\xi_{q'_i}]$, $q'_i \in \mathbb{Z}^+$ for $i \in [3]$, and ξ_m is the m-th root of unity of some irreducible polynomial $f(X) \in \mathbb{Z}[X]$ for $m \in \mathbb{Z}^+$. Let $\tilde{K} = \mathbb{Q}[\xi_{qq'_0q'_1}]$ and $\tilde{K}_2 = \mathbb{Q}[\xi_{qq'_0q'_1q'_2}]$. Then we have isomorphisms: $\tilde{K} \cong K \otimes K_0 \otimes K_1$ and $\tilde{K}_2 \cong K \otimes K_0 \otimes K_1 \otimes K_2$. Let $\tilde{\mathcal{R}}$, $\tilde{\mathcal{R}}_2$, $\mathcal{R}$ and $\mathcal{R}_i$ be the rings of integers of $\tilde{K}$, $\tilde{K}_2$, K and K_i for $i \in [3]$, respectively. Then, $\tilde{\mathcal{R}} \cong \mathcal{R} \otimes \mathcal{R}_0 \otimes \mathcal{R}_1$ and $\tilde{\mathcal{R}}_2 \cong \mathcal{R} \otimes \mathcal{R}_0 \otimes \mathcal{R}_1 \otimes \mathcal{R}_2$.

In addition, let $q_i = \phi(q'_i)$ for $i \in [3]$ where ϕ is the Euler function. Let $\{p_i\}_{i\in[3]}$ and q be different odd prime numbers, where p_i is a small positive integer of constant size for $i \in [3]$. Set $q'_i = p_i^{d_i}$ such that $\mathsf{gcd}(Q, q'_i) = 1$ where $d_i \in \mathbb{Z}^+$ for $i \in [3]$. Set $q_0 \approx q_1 \approx q_2$, $u = \min_{i\in[3]}\{q_i\}$, $\tilde{N} = \phi(qq'_0q'_1)$, $\tilde{N}_2 = \phi(qq'_0q'_1q'_2)$, $\mathcal{R} = \mathbb{Z}[X]/(\Phi_q(X))$ where $\Phi_q(X) = \sum_{i=0}^{q-1} X^i$.

For $i \in [3]$, denote $B_i = \{\mathsf{r}_{i,0}, \ldots, \mathsf{r}_{i,q_i-1}\}$ and $B_i^\vee = \{\mathsf{r}^\vee_{i,0}, \ldots, \mathsf{r}^\vee_{i,q_i-1}\}$ as the bases of $\mathcal{R}_i$ and $\mathcal{R}_i^\vee$, respectively. Set B_i as the conjugate of the powerful basis of $\mathcal{R}_i$ (i.e. $\mathsf{r}_{i,j} = \xi_{q'_i}^{-j}$ for $j \in [q_i]$), and set $B_i^\vee$ as the decoding basis of $\mathcal{R}_i^\vee$ (i.e. the dual of B_i) for $i \in [3]$. Then, based on [24,19], $\|\mathsf{r}_{i,j}\| = 1$ and $\|\mathsf{r}^\vee_{i,j}\| \le \frac{2(p_i-1)}{q'_i}$ for $j \in [q_i]$ and $i \in [3]$. Define $\tilde{\mathsf{r}}_{0,j} = \mathsf{r}^\vee_{0,j}\mathsf{r}_{1,j}$ and $\tilde{\mathsf{r}}_{1,j} = \mathsf{r}_{0,j}\mathsf{r}^\vee_{1,j}$ for $j \in [u]$. For the trace functions $\mathsf{Tr}_{\tilde{K}_2/\tilde{K}} : \tilde{K}_2 \to \tilde{K}$ and $\mathsf{Tr}_{\tilde{K}/\tilde{K}_k} : \tilde{K} \to \tilde{K}_k$ where $\tilde{K}_k = K \otimes K_k$ for $k \in [2]$, it holds that $\mathsf{Tr}_{\tilde{K}_2/\tilde{K}}(\mathsf{r}_{2,i} \cdot \mathsf{r}^\vee_{2,j}) = \delta_{i,j}$ for $i, j \in [q_2]$, and $\mathsf{Tr}_{\tilde{K}/\tilde{K}_{1-k}}(\mathsf{r}_{k,i} \cdot \mathsf{r}^\vee_{k,j}) = \delta_{i,j}$ for $i, j \in [q_k]$ and $k \in [2]$ where $\delta_{i,j} = 1$ if $i = j$, and $\delta_{i,j} = 0$ if $i \neq j$.

Packing. By the packing method of [19,20], one can pack multiple elements in $\mathcal{R}$ into an element in a bigger ring based on tensor rings. Besides, one can extend Pack to a matrix over $\mathcal{R}$ by applying it to each element of the matrix.

- $\mathsf{Pack}(\{x_i\}_{i\in[u]}, M)$: Given polynomials $\{x_i \in \mathcal{R}\}_{i\in[u]}$ and a mode $M \in [4]$, pack the polynomials as $x' = \sum_{i=0}^{u-1} x_i \mathsf{r}_{M,i}$ if $M \in \{0,1\}$ and $x' = \sum_{i=0}^{u-1} x_i \tilde{\mathsf{r}}_{M-2,i}$ if $M \in \{2,3\}$. Output x'. We call x' as the encoding of $\{x_i\}_{i\in[u]}$ with the mode M.
- Addition: Given (x', M) and (y', M) where $x' \leftarrow \mathsf{Pack}(\{x_i\}_{i\in[u]}, M)$, $y' \leftarrow \mathsf{Pack}(\{y_i\}_{i\in[u]}, M)$, and $M \in [4]$, compute $z_{\mathsf{add}} \leftarrow x' + y'$, and then output (z_{add}, M). Then z_{add} is the encoding of $\{x_i + y_i\}_{i\in[u]}$ with the mode M.
- Multiplication: Given (x', M_0) and (y', M_1) where $x' \leftarrow \mathsf{Pack}(\{x_i\}_{i\in[u]}, M_0)$, $y' \leftarrow \mathsf{Pack}(\{y_i\}_{i\in[u]}, M_1)$, and $M_0, M_1 \in [4]$, compute $z_{\mathsf{mul}} \leftarrow \mathsf{Tr}'(x \cdot y)$ and then output (z_{mul}, M), where $\mathsf{Tr}' = \mathsf{Tr}_{\tilde{K}/\tilde{K}_1}$ and $M = 1$ if $M_0 = 0, M_1 = 2$ or vice versa, $\mathsf{Tr}' = \mathsf{Tr}_{\tilde{K}/\tilde{K}_0}$ and $M = 0$ if $M_0 = 1, M_1 = 3$ or vice versa. Then z_{mul} is the encoding of $\{x_i \cdot y_i\}_{i\in[u]}$ with the mode M.

Trace. Given an RLWE ciphertext of μ, one can compute an RLWE ciphertext of $\mathsf{Tr}_{\tilde{K}/\tilde{K}_i}(\mu)$ for $i \in [2]$ by the trace evaluation algorithm of [19]. In this algorithm, assume that there are many intermediate fields between $\tilde{K}$ and $\tilde{K}_i$ for $i \in [2]$. Specifically, assume that we have tower structures: $\tilde{K}_i = T_{i,t_i-1} \subset T_{i,t_i-2} \subset \cdots \subset T_{i,0} = \tilde{K}$ for $i \in [2]$. Then $\mathsf{Tr}_{\tilde{K}/\tilde{K}_i} = \mathsf{Tr}_{T_{i,t_i-2}/T_{i,t_i-1}} \circ \mathsf{Tr}_{T_{i,t_i-3}/T_{i,t_i-2}} \circ \cdots \circ \mathsf{Tr}_{T_{i,0}/T_{i,1}}$ for $i \in [2]$. And we denote the degrees as $\tau_{i,j} = [T_{i,j} : T_{i,j+1}]$ and set $\tau_{i,j} = O(1)$ for $j \in [t_i - 1]$ and $i \in [2]$. Then $q_{1-i} = \prod_{j=0}^{t_i-2} \tau_{i,j}$ and $\sum_{j=0}^{t_i-2} \tau_{i,j} = O(\log q_{1-i})$ for $i \in [2]$. Let $\boldsymbol{EK} = \{\boldsymbol{EK}_b \in \mathsf{REG}^{Q}_{z,\tilde{R}}(zq'^{-1}_{1-b})\}_{b\in[2]}$ and $\boldsymbol{TK} = \{\boldsymbol{TK}_\theta \in \mathsf{REG}^{Q}_{z,\tilde{R}}(\theta(z))\}_{\theta\in L}$ be the trace keys with errors sub-Gaussian with parameter E, where $L = \cup_{i=0}^{1}(\cup_{j=0}^{t_i-2}\mathsf{Gal}(T_{i,j}/T_{i,j+1}))$.

In this paper, we replace the external product between an RGSW ciphertext $\boldsymbol{C} = (\boldsymbol{C}_0, \boldsymbol{C}_1)$ and an RLWE ciphertext $(0, a)$ in the trace evaluation algorithm of [19] with the external product between the REG ciphertext $\boldsymbol{C}_1$ and the polynomial a, as shown below. Note that this replacement essentially does not change the algorithm of [19]. The correctness is given in Lemma 5.

- $\mathsf{Trace}(\boldsymbol{c}, M, \boldsymbol{EK}, \boldsymbol{TK})$ Given an RLWE ciphertext $\boldsymbol{c} = (a, b)$, a mode $M \in [2]$, and trace keys $\boldsymbol{EK}$ and $\boldsymbol{TK}$, compute $\boldsymbol{c}' \leftarrow \boldsymbol{EK}_M \odot (q'_{1-M} \cdot a)$ and set $\boldsymbol{d}_0 = \boldsymbol{c}'$. Evaluate $\boldsymbol{d}_{j+1} \leftarrow \sum_{\theta\in\mathsf{Gal}(T_{M,j}/T_{M,j+1})} \mathsf{Aut}(\boldsymbol{d}_j, \theta, \boldsymbol{TK}_\theta)$ iteratively for $j = 0, \ldots, t_M - 2$. Compute $\boldsymbol{c}_{\mathsf{tr}} \leftarrow (0, \mathsf{Tr}_{\tilde{K}/\tilde{K}_M}(b)) - \boldsymbol{d}_{t_M-1}$ and output it.

Lemma 5 ([19]). *Let $M \in [2]$. Let $\boldsymbol{c}$ be an RLWE ciphertext such that $\boldsymbol{c}^\top \boldsymbol{z}' = \mu + e \pmod{Q}$ where $\boldsymbol{z}' = (-z, 1)$, $\boldsymbol{c} \in (\mathcal{R} \otimes \mathcal{R}_0 \otimes \mathcal{R}_1^\vee)^2$ if $M = 0$, and $\boldsymbol{c} \in (\mathcal{R} \otimes \mathcal{R}_0^\vee \otimes \mathcal{R}_1)^2$ if $M = 1$. If $\boldsymbol{c}_{\mathsf{tr}} \leftarrow \mathsf{Trace}(\boldsymbol{c}, M, \boldsymbol{EK}, \boldsymbol{TK})$, then $\boldsymbol{c}_{\mathsf{tr}}$ is an RLWE ciphertext such that $\boldsymbol{c}_{\mathsf{tr}}^\top \boldsymbol{z}' = \mathsf{Tr}_{\tilde{K}/\tilde{K}_M}(\mu + e) + e_{\mathsf{tr}} \pmod{Q}$ where e_{tr} is sub-Gaussian with parameter $O(uE\sqrt{\tilde{N}\ell})$.*

Amortized External Product. Given a packed RGSW ciphertext and a packed RLWE ciphertext, one can use the amortized external product of [19], constructed by Trace, to compute the multiplication of these ciphertexts. The procedure and correctness are shown below.

- $\mathsf{BatchMul}((\boldsymbol{D}, M_0), (\boldsymbol{d}, M_1), \boldsymbol{EK}, \boldsymbol{TK})$: Given a packed RGSW ciphertext $\boldsymbol{D}$, a packed RLWE ciphertext $\boldsymbol{d}$, their modes M_0 and M_1, and trace keys $\boldsymbol{EK}$ and $\boldsymbol{TK}$, compute $\boldsymbol{d}_{\mathsf{mul}} \leftarrow \boldsymbol{D} \boxtimes \boldsymbol{d}$ and $M \leftarrow 1 - M_1$, and then evaluate $\boldsymbol{d}_{\mathsf{out}} \leftarrow \mathsf{Trace}(\boldsymbol{d}_{\mathsf{mul}}, M, \boldsymbol{EK}, \boldsymbol{TK})$. Output $\boldsymbol{d}_{\mathsf{out}}$.

Lemma 6 ([19]). *For $i \in [u]$, let $\boldsymbol{C}_i \in \mathsf{RGSW}^{Q}_{z,\mathcal{R}}(\mu_i)$ with an error $\boldsymbol{e}_i$ and let $\bar{\boldsymbol{c}}_i \in \mathsf{RLWE}^{Q,\Delta}_{z,\mathcal{R}}(\bar{\mu}_i)$ with an error $\bar{e}_i$. Let $\boldsymbol{D} \leftarrow \mathsf{Pack}(\{\boldsymbol{C}_i\}_{i\in[u]}, M_0)$ and $\boldsymbol{d} \leftarrow \mathsf{Pack}(\{\bar{\boldsymbol{c}}_i\}_{i\in[u]}, M_1)$ for $M_0 \in \{2,3\}, M_1 \in \{0,1\}$. Use $\boldsymbol{EK}$ and $\boldsymbol{TK}$ defined above. If $\boldsymbol{d}_{\mathsf{out}} \leftarrow \mathsf{BatchMul}((\boldsymbol{D}, M_0), (\boldsymbol{d}, M_1), \boldsymbol{EK}, \boldsymbol{TK})$ and $M_0 - 2 = M_1$. Then $\boldsymbol{d}_{\mathsf{out}}^{\top}\boldsymbol{z}' = \Delta \sum_{i=0}^{u-1} \mu_i \bar{\mu}_i \mathsf{r}_{M,i} + e'_{\mathsf{mul}} + \mathsf{Tr}_{\tilde{K}/\tilde{K}_M}(\mu_D e_d) \pmod{Q}$, where $M = 1 - M_1$, $e'_{\mathsf{mul}} = \mathsf{Tr}_{\tilde{K}/\tilde{K}_M}(\boldsymbol{e}_D^{\top}\mathsf{G}^{-1}(\boldsymbol{d})) + e_{\mathsf{tr}}$, $\mu_D = \sum_{i=0}^{u-1} \mu_i \tilde{\mathsf{r}}_{M_1,i}$, $\boldsymbol{e}_D = \sum_{i=0}^{u-1} \boldsymbol{e}_i \tilde{\mathsf{r}}_{M_1,i}$, $e_d = \sum_{i=0}^{u-1} \bar{e}_i \mathsf{r}_{M_1,i}$, and e_{tr} is sub-Gaussian with parameter $O(uE\sqrt{\tilde{N}\ell})$.*

According to [19], there is a useful property about the error e_k of the amortized external products of an RLWE ciphertext and k RGSW ciphertexts for $k \geq 2$. Let $e_j = e'_{\mathsf{mul},j} + H_j(\mu_{D,j} \cdot e_{j-1})$, where $H_j = \mathsf{Tr}_{\tilde{K}/\tilde{K}_{1-j'}}$, $\mu_{D,j} = \sum_{i=0}^{u-1} \mu_{i,j} \tilde{\mathsf{r}}_{j',i}$, $j' = [j]_2^+$, and $\mu_{i,j}$ is a monomial for $i \in [u]$ and $j \in \{1, \ldots, k\}$. Let $\tilde{e}_j = e'_{\mathsf{mul},j} + H_j(\mu_{D,j} \cdot e'_{\mathsf{mul},j-1})$ for $j \in \{1, \ldots, k\}$ and $\tilde{e}_0 = e_0$. Without loss of generality, we only consider the case where k is even. Let

$$T^{2j}(e) = \begin{cases} \mathsf{Tr}_{\tilde{K}/\tilde{K}_1}(\mu_{D,k} \cdot \mathsf{Tr}_{\tilde{K}/\tilde{K}_0}(\mu_{D,k-1} \cdot e)) & if\ j = 1 \\ T^{2j-2}(\mathsf{Tr}_{\tilde{K}/\tilde{K}_1}(\mu_{D,k-2j+2} \cdot \mathsf{Tr}_{\tilde{K}/\tilde{K}_0}(\mu_{D,k-2j+1} \cdot e))) & 2 \leq j \leq \frac{k}{2} \end{cases}.$$

Then $e_k = \tilde{e}_k + \sum_{j=1}^{\frac{k}{2}} T^{2j}(\tilde{e}_{k-2j})$ and $T^{2j}(e)$ is sub-Gaussian with parameter $O(u^2E)$ when e is sub-Gaussian with parameter E.

Unpacking. Given a ciphertext that packs and encrypts $\{x_i\}_{i\in[u]}$, one can use the unpacking method of [19], built by Trace, to obtain a ciphertext of x_i for each $i \in [u]$. The procedure and correctness are given below.

- $\mathsf{UnPack}(\boldsymbol{d}, M, \boldsymbol{EK}, \boldsymbol{TK})$: Given a packed RLWE ciphertext $\boldsymbol{d}$, a mode $M \in [2]$, and trace keys $\boldsymbol{EK}$ and $\boldsymbol{TK}$, compute $\boldsymbol{c}'_i \leftarrow \mathsf{Trace}(\boldsymbol{d} \cdot \mathsf{r}^{\vee}_{M,i}, 1-M, \boldsymbol{EK}, \boldsymbol{TK})$ for $i \in [u]$, and then output $\{\boldsymbol{c}'_i\}_{i\in[u]}$.

Lemma 7 ([19]). *Let $\boldsymbol{c}_i \in \mathsf{RLWE}^{Q,\Delta}_{z,\mathcal{R}}(\mu_i)$ with an error e_i for $i \in [u]$. Let $\boldsymbol{d} \leftarrow \mathsf{Pack}(\{\boldsymbol{c}\}_{i\in[u]}, M)$ where $M \in [2]$. Use $\boldsymbol{EK}$ and $\boldsymbol{TK}$ defined above. Then $\mathsf{UnPack}(\boldsymbol{d}, M, \boldsymbol{EK}, \boldsymbol{TK})$ outputs $\{\boldsymbol{c}'_i \in \mathsf{RLWE}^{Q,\Delta}_{z,\mathcal{R}}(\mu_i)\}_{i\in[u]}$ where $\boldsymbol{c}_i'^{\top}\boldsymbol{z}' = \Delta\mu_i + e_i + e_{\mathsf{tr},i} \pmod{Q}$ and $e_{\mathsf{tr},i}$ is sub-Gaussian with parameter $O(uE\sqrt{\tilde{N}\ell})$.*

3 Amortized Functional Bootstrapping

In this section, we propose a new amortized functional bootstrapping algorithm $\mathsf{BatchFB}$ whose amortized complexity is $\tilde{O}(1)$ for any public or encrypted function. We first introduce an extended amortized homomorphic automorphism algorithm $\mathsf{BatchAut}$ and use it to build an improved homomorphic inverse NTT algorithm $\mathsf{HomINTT}$. Then we use $\mathsf{HomINTT}$ to construct the algorithm $\mathsf{BatchFB}$, analyze its output errors and complexity, and compare it with previous works.

3.1 Amortized Homomorphic Automorphism

Given automorphisms $\{\theta_i\}_{i\in[u]}$ of $\mathcal{R}$ and ciphertexts $\{\boldsymbol{c}_i=(a_i,b_i)\}_{i\in[u]}$ of $\{\mu_i\in\mathcal{R}\}_{i\in[u]}$ under the secret key z, we attempt to compute a ciphertext of a plaintext that encodes $\{\theta_i(\mu_i)\}_{i\in[u]}$ by batching. Here, we consider the following automorphisms. Define $\psi_k:\mathcal{R}\to\mathcal{R}$ as an automorphism of $\mathcal{R}$ such that $a(X)\mapsto a(X^k)$ for $k\in\mathbb{Z}_q^*$, where $\mathbb{Z}_q^*$ is composed of all reversible elements in $\mathbb{Z}_q$.

The idea of our amortized homomorphic automorphism algorithm $\mathsf{BatchAut}$ is as follows. According to Aut, computing $\theta_i(b_i)-\theta_i(a_i)\cdot\theta_i(z)$ homomorphically is necessary to accomplish this task with homomorphic automorphisms. Based on Lemma 5, one can use packing and trace functions to batch multiple homomorphic multiplications and subtractions. But as illustrated in $\mathsf{HomINTT}$ constructed later, we want $\mathsf{BatchAut}$ to output an RLWE ciphertext of the plaintext $\sum_{i=0}^{u-1}\theta_i(\mu_i)\tilde{\mathsf{r}}_{M,i}$ for $M\in[2]$. For this purpose, we need to use the tensor ring $\tilde{R}_2$ and evaluate the trace function $\mathsf{Tr}_{\tilde{K}_2/\tilde{K}}$ homomorphically.

Similar to [19,20], to reduce the computational cost, assume that we have a tower structure: $\tilde{K}=T_{2,t_2-1}\subset T_{2,t_2-2}\subset\cdots\subset T_{2,0}=\tilde{K}_2$. Then $\mathsf{Tr}_{\tilde{K}_2/\tilde{K}}=\mathsf{Tr}_{T_{2,t_2-2}/T_{2,t_2-1}}\circ\mathsf{Tr}_{T_{2,t_2-3}/T_{2,t_2-2}}\circ\cdots\circ\mathsf{Tr}_{T_{2,0}/T_{2,1}}$. Set the degrees as $\tau_{2,j}=[T_{2,j}:T_{2,j+1}]=O(1)$ for $j\in[t_2-1]$. Then $q_2=\prod_{j=0}^{t_2-2}\tau_{2,j}$ and $\sum_{j=0}^{t_2-2}\tau_{2,j}=O(\log q_2)$. As for the keys used in $\mathsf{BatchAut}$, let $\boldsymbol{EK}=\{\boldsymbol{EK}_b\in\mathsf{REG}^{Q}_{z,\tilde{\mathcal{R}}_2}(z(q_b'q_2')^{-1})\}_{b\in[2]}$, $\boldsymbol{TK}=\{\boldsymbol{TK}_\theta\in\mathsf{REG}^{Q}_{z,\tilde{\mathcal{R}}_2}(\theta(z))\}_{\theta\in L}$ where $L=\cup_{j=0}^{t_2-2}\mathsf{Gal}(T_{2,j}/T_{2,j+1})$, and $\boldsymbol{AK}=\{\boldsymbol{AK}_{\psi_k}\in\mathsf{REG}^{Q}_{z,\mathcal{R}}(\psi_k(z))\}_{k\in\mathbb{Z}_q^*}$. Let their errors be sub-Gaussian with parameter E. The detailed procedure of $\mathsf{BatchAut}$ is shown in Algorithm 1, and its correctness is demonstrated in Lemma 8.

Lemma 8. *For $i\in[u]$, let $\boldsymbol{c}_i\in\mathsf{RLWE}^{Q,\Delta}_{z,\mathcal{R}}(\mu_i)$ with an error e_i. and let $\theta_i\in\{\psi_k\}_{k\in\mathbb{Z}_q^*}$ be an automorphism of $\mathcal{R}$. If $\boldsymbol{c}'\leftarrow\mathsf{BatchAut}(\{\boldsymbol{c}_i\}_{i\in[u]},\{\theta_i\}_{i\in[u]},\boldsymbol{AK},M,\boldsymbol{EK},\boldsymbol{TK})$ for $M\in\{2,3\}$, then $\boldsymbol{c}'^{\top}\boldsymbol{z}'=\sum_{i=0}^{u-1}\theta_i(\Delta\mu_i+e_i)\tilde{\mathsf{r}}_{M-2,i}+e'\pmod Q$ where $\boldsymbol{z}'=(-z,1)$ and e' is sub-Gaussian with parameter $O(uE\sqrt{\tilde{N}_2\ell})$.*

Proof. Assume $M=2$. Let $\boldsymbol{e}_{\mathsf{ak},i}$ be the error of $\boldsymbol{AK}_{\theta_i}$ for $i\in[u]$. By steps 3 to 4 and Lemma 1, $\boldsymbol{c}_{\mathsf{mul}}^{\top}\boldsymbol{z}'=\alpha\mu_{\mathsf{AK}}+e_{\mathsf{mul}}(\bmod Q)$ where $\mu_{\mathsf{AK}}=\sum_{i=0}^{u-1}\theta_i(z)\mathsf{r}_{1,i}\mathsf{r}^{\vee}_{2,i}$, and by step 5, $\boldsymbol{d}_0^{\top}\boldsymbol{z}'=za_{\mathsf{mul}}+e'_{\mathsf{mul}}\pmod Q$ where e_{mul} and e'_{mul} are sub-Gaussian with parameter $O(E\sqrt{\tilde{N}_2\ell})$. By steps 6 to 8 and Lemma 2, $\boldsymbol{c}_{\mathsf{tr}}'^{\top}\boldsymbol{z}'=\sum_{i=0}^{u-1}\theta_i(a_i\cdot z)\cdot\zeta_i\mathsf{r}_{1,i}+e'\pmod Q$ where e' is sub-Gaussian with parameter $O(uE\sqrt{\tilde{N}_2\ell})$. By

Algorithm 1 BatchAut

Input: RLWE ciphertexts $\{\boldsymbol{c}_i = (a_i, b_i)\}_{i\in[u]}$, $\{\theta_i\}_{i\in[u]}$, $\boldsymbol{AK}$, $M \in \{2, 3\}$, $\boldsymbol{EK}$, $\boldsymbol{TK}$
Output: an RLWE ciphertext $\boldsymbol{c}'$

1: $(\zeta_i)_{i\in[u]} \leftarrow (\mathsf{r}^{\vee}_{M-2,i})_{i\in[u]}$
2: $\alpha \leftarrow \sum_{i=0}^{u-1}(\theta_i(a_i)\cdot\zeta_i)\cdot \mathsf{r}_{2,i}$, $\beta \leftarrow \sum_{i=0}^{u-1}(\theta_i(b_i)\cdot\zeta_i)\cdot \mathsf{r}_{1-[M]_2^+,i}$
3: $\boldsymbol{C} \leftarrow \sum_{i=0}^{u-1} \boldsymbol{AK}_{\theta_i}\cdot(\mathsf{r}_{1-[M]_2^+,i}\mathsf{r}^{\vee}_{2,i})$
4: $\boldsymbol{c}_{\mathsf{mul}} = (a_{\mathsf{mul}}, b_{\mathsf{mul}}) \leftarrow \boldsymbol{C}\odot\alpha$
5: $\boldsymbol{d}_0 \leftarrow \boldsymbol{EK}_{M-2}\odot(q'_{M-2}\cdot q'_2\cdot a_{\mathsf{mul}})$
6: **for** $i = 0$ to $t_2 - 2$ **do**
7: $\quad \boldsymbol{d}_{i+1} \leftarrow \sum_{\theta\in\mathsf{Gal}(T_{2,i}/T_{2,i+1})} \mathsf{Aut}(\boldsymbol{d}_i, \theta, \boldsymbol{TK}_\theta)$
8: $\boldsymbol{c}_{\mathsf{tr}} \leftarrow (0, \mathsf{Tr}_{\tilde{K}_2/\tilde{K}}(b_{\mathsf{mul}})) - \boldsymbol{d}_{t_2-1}$
9: $\boldsymbol{c}' \leftarrow (0, \beta) - \boldsymbol{c}_{\mathsf{tr}}$

step 9, $\boldsymbol{c}'^{\top}\boldsymbol{z}' = \beta - \sum_{i=0}^{u-1}\theta_i(a_i\cdot z)\cdot\zeta_i\mathsf{r}_{1,i} - e' = \sum_{i=0}^{u-1}\theta_i(\Delta\mu_i + e_i)\tilde{\mathsf{r}}_{0,i} - e' \pmod Q$. The result can be verified similarly when $M = 3$.

If we only utilize Aut to homomorphically evaluate $\{\theta_i(\mu_i)\}_{i\in[u]}$ like in [15,27], then the number of the operation $\odot$ needed in this evaluation is $O(u)$. However, by Algorithm 1, the number of $\odot$ is only $O(\log u)$. In [20], Liu and Wang also proposed an amortized homomorphic automorphism algorithm, but their algorithm requires performing the same automorphism on the input ciphertexts and is only suitable for a mode $M \in [2]$. So, our algorithm is more powerful and has a wider range of applications.

3.2 Homomorphic Inverse NTT

Homomorphic inverse NTT is the core of amortized functional bootstrapping algorithms. Below, we first review the NTT technique and then use our extended amortized homomorphic automorphism algorithm to construct an improved homomorphic inverse NTT algorithm.

NTT. Let N be a power of two and $N = r^\rho$ where $r \le u$ and $\rho = O(1)$. Let $\hat{\mathcal{R}} = \mathbb{Z}[X]/(\Phi_{2N}(X))$ where $\Phi_{2N}(X) = X^N + 1$. As the prime number q satisfies $q = 1 \pmod{2N}$, there exists a primitive $2N$-root of unity $\omega_{2N} \in \mathbb{Z}_q$. For $k \in \{1, \ldots, \rho\}$, define the r^k-th root of unity in $\mathbb{Z}_q$ as $\omega_{r^k} = [\omega_{2N}^{(2N)/r^k}]_q$. For $a = \sum_{i=0}^{N-1} a_i X^i \in \hat{\mathcal{R}}_q$, $\mathsf{NTT}(a) = \boldsymbol{a}' = (a'_i)_{i\in[N]}$ where $a'_i \leftarrow [\sum_{j=0}^{N-1} a_j\omega_{2N}^{(2i+1)j}]_q$ for $i \in [N]$, $\mathsf{INTT}(\boldsymbol{a}') = a'' = \sum_{k=0}^{N-1} a''_k X^k$ where $a''_k \leftarrow [N^{-1}\cdot\sum_{i=0}^{N-1} a'_i\omega_{2N}^{-(2i+1)k}]_q$ for $k \in [N]$. Then, $a'' = a$. For $b \in \hat{\mathcal{R}}_q$ and $(b'_i)_{i\in[N]} = \mathsf{NTT}(b)$, if $\boldsymbol{c}' = (a'_i\cdot b'_i)_{i\in[N]}$, then $a\cdot b = \mathsf{INTT}(\boldsymbol{c}')$. Similar to [15], we use the radix-r inverse NTT algorithm that recursively splits the N-dimensional input into r vectors of dimension $\frac{N}{r}$ to perform inverse NTTs on shorter vectors, obtaining the complexity $O(N\log N)$.

Algorithm 2 HomINTT

Input: REG ciphertexts $\{\boldsymbol{C}_i\}_{i\in[r^{\rho-\rho'+1}]}$, $\rho' \in \mathbb{Z}^+$, $\rho \in \mathbb{Z}^+$, $\boldsymbol{AK}, \overline{\boldsymbol{EK}}, \overline{\boldsymbol{TK}}, \boldsymbol{SwK}$
Output: REG or RLWE ciphertexts $\{\boldsymbol{C}'_j\}_{j\in[r^{\rho-\rho'+1}]}$

if $\rho' \neq \rho$ **then**
  for $i = 0$ to $r-1$ **do**
    $\hat{\boldsymbol{B}} \leftarrow \{\boldsymbol{C}_{i+r\cdot 0}, \boldsymbol{C}_{i+r\cdot 1}, \ldots, \boldsymbol{C}_{i+r\cdot(r^{\rho-\rho'}-1)}\}$
    $\{\boldsymbol{B}_{i,k_0}\}_{k_0\in[r^{\rho-\rho'}]} \leftarrow \mathsf{HomINTT}(\hat{\boldsymbol{B}}, \rho'+1, \rho, \boldsymbol{AK}, \overline{\boldsymbol{EK}}, \overline{\boldsymbol{TK}}, \boldsymbol{SwK})$
else
  $\{\boldsymbol{B}_{i,0}\}_{i\in[r]} \leftarrow \{\boldsymbol{C}_i\}_{i\in[r]}$
for $k_0 = 0$ to $r^{\rho-\rho'}-1$ **do**
  if $\rho' = 1$ **then**
    $\boldsymbol{d}'_0 \leftarrow \mathsf{Pack}(\{T_j\}_{j\in[r]}, 0)$ where $T_j \leftarrow \boldsymbol{B}_{0,k_0,0}$ for $j \in [r]$
  else
    $\boldsymbol{d}'_0 \leftarrow \mathsf{Pack}(\{T_j\}_{j\in[r]}, 0)$ where $T_j \leftarrow \boldsymbol{B}_{0,k_0}$ for $j \in [r]$
  for $i = 1$ to $r-1$ **do**
    $M \leftarrow 3 - [i]_2^+$, $\tilde{T} \leftarrow \{T_{k_1}\}_{k_1\in[r]}$ where $T_{k_1} \leftarrow \boldsymbol{B}_{i,k_0}$ for $k_1 \in [r]$
    $\boldsymbol{C}'_i \leftarrow \mathsf{BatchAut}(\tilde{T}, \{\theta_{k_1}\}_{k_1\in[r]}, \{\boldsymbol{AK}_{\theta_{k_1}}\}_{k_1\in[r]}, M, \boldsymbol{EK}', \boldsymbol{TK}')$ where $\theta_{k_1} = \psi_{\gamma_{i,k_0,k_1}}$, $\gamma_{i,k_0,k_1} = [\omega_{r^{\rho-\rho'+1}}^{-ik_0}\omega_r^{-ik_1}]_q$
    $\boldsymbol{D}_i \leftarrow \mathsf{Switch}(\boldsymbol{C}'_i, \boldsymbol{SwK})$
    $\boldsymbol{d}'_i \leftarrow \mathsf{BatchMul}((\boldsymbol{D}_i, M), (\boldsymbol{d}'_{i-1}, 1-[i]_2^+), \boldsymbol{EK}, \boldsymbol{TK})$
  $\{\boldsymbol{C}'_{k_0+k_1\cdot r^{\rho-\rho'}}\}_{k_1\in[r]} \leftarrow \mathsf{Unpack}(\boldsymbol{d}'_{r-1}, 1-[r]_2^+, \boldsymbol{EK}, \boldsymbol{TK})$
if $\rho' = 1$ **then**
  for $j = 0$ to $N-1$ **do**
    $\boldsymbol{C}'_j \leftarrow \mathsf{Aut}(\boldsymbol{C}'_j, \theta, \boldsymbol{AK}_\theta)$ where $\theta = \psi_{[r^{-\rho}\cdot\omega_{2N}^{-j}]_q}$

Homomorphic Inverse NTT. We improve the homomorphic inverse NTT algorithm of [15] to perform radix-r inverse NTT on ciphertexts homomorphically. While prior efforts like [26,15,27,20] have not considered a test polynomial v encoding a function throughout this process, our algorithm HomINTT does. The keys used in HomINTT are as follows. Let $\boldsymbol{AK} = \{\boldsymbol{AK}_{\psi_k} \in \mathsf{REG}^Q_{z,\mathcal{R}}(\psi_k(z))\}_{k\in\mathbb{Z}_q^*}$, $\boldsymbol{SwK} \in \mathsf{REG}^Q_{z,\tilde{R}}(z^2)$, $\overline{\boldsymbol{EK}} = \{\boldsymbol{EK}, \boldsymbol{EK}'\}$, $\overline{\boldsymbol{TK}} = \{\boldsymbol{TK}, \boldsymbol{TK}'\}$ where $\boldsymbol{EK} = \{\boldsymbol{EK}_b \in \mathsf{REG}^Q_{z,\tilde{\mathcal{R}}}(zq'^{-1}_{1-b})\}_{b\in[2]}$, $\boldsymbol{TK} = \{\boldsymbol{TK}_\theta \in \mathsf{REG}^Q_{z,\tilde{\mathcal{R}}}(\theta(z))\}_{\theta\in L}$, $\boldsymbol{EK}' = \{\boldsymbol{EK}'_b \in \mathsf{REG}^Q_{z,\tilde{\mathcal{R}}_2}(zq'^{-1}_b q'^{-1}_2)\}_{b\in[2]}$, $\boldsymbol{TK}' = \{\boldsymbol{TK}'_\theta \in \mathsf{REG}^Q_{z,\tilde{\mathcal{R}}_2}(\theta(z))\}_{\theta\in L'}$, $L = \cup_{i=0}^{1}(\cup_{j=0}^{t_i-2}\mathsf{Gal}(T_{i,j}/T_{i,j+1}))$, $L' = \cup_{j=0}^{t_2-2}\mathsf{Gal}(T_{2,j}/T_{2,j+1})$. Let the errors of these keys be sub-Gaussian with parameter E_{ks}.

The main idea of HomINTT (see Algorithm 2) is shown below. Input $N = r^\rho$ REG ciphertexts $\{\boldsymbol{C}_i\}_{i\in[N]}$ of $\{\mu_i\}_{i\in[N]}$ with an error sub-Gaussian with parameter B_i where $\mu_i = X^{\tilde{\varphi}_i}$ and $B_i = E_0$ for $i \in [N]\setminus\{0\}$, $\mu_0 = vX^{\tilde{\varphi}_0}$, $B_0 = E_1$, $v \in \mathcal{R}$, and $\tilde{\varphi}_i \in \mathbb{Z}_q$ for $i \in [N]$. First, divide them into r parts iteratively. So the algorithm requires ρ layers of iterations. In the ρ'-th layer ($1 \le \rho' \le \rho$), it is necessary to perform homomorphic inverse NTT on $r^{\rho-\rho'}$ REG ciphertexts for each part. This involves the homomorphic computation

of an REG ciphertext of $X^{a'_j}$, given constants $\{\gamma_{i,k_0,k_1}\}$ and REG ciphertexts $\{\boldsymbol{B}_{i,k_0}\}$ of $\{X^{b_{i,k_0}}\}$ with errors sub-Gaussian with parameter $E_{\rho'+1}$ where $a'_j = \sum_{i=0}^{r-1} b_{i,k_0} \cdot \gamma_{i,k_0,k_1}$, $j = k_0 + k_1 r^{\rho-\rho'}$, $\gamma_{i,k_0,k_1} = [\omega_{r^{\rho-\rho'+1}}^{-ik_0} \omega_r^{-ik_1}]_q$, $b_{i,k_0} \in \mathbb{Z}_q$, for $i \in [r], k_0 \in [r^{\rho-\rho'}], k_1 \in [r]$. For each k_0, to compute the ciphertext of $X^{a'_j}$, we first copy r times $\boldsymbol{B}_{0,k_0}$ and then pack the copies, obtaining an REG ciphertext $\boldsymbol{d}'_0$. Then, we copy r times $\boldsymbol{B}_{i,k_0}$ and then perform our extended amortized homomorphic automorphism algorithm on the copies to calculate an REG ciphertext $\boldsymbol{C}'_i$ which encrypts a plaintext encoding $\{X^{b_{i,k_0} \cdot \gamma_{i,k_0,k_1}}\}_{k_1 \in [r]}$ and has an error sub-Gaussian with parameter $E_C = O(E_{\rho'+1}) + O(uE_{\mathsf{ks}}\sqrt{\tilde{N}_2 \ell})$ based on Lemma 8, for $i \in \{1, \ldots, r-1\}$. Next, by scheme switching, we convert each $\boldsymbol{C}'_i$ into an RGSW ciphertext $\boldsymbol{D}_i$ whose error is sub-Gaussian with parameter $E_D = O(\sqrt{h} E_C) + O(\sqrt{\tilde{N}\ell} E_{\mathsf{ks}})$ based on Lemma 3. Moreover, by amortized external products, we use $\{\boldsymbol{D}_i\}$ and $\boldsymbol{d}'_0$ to calculate an REG ciphertext $\boldsymbol{d}'_{r-1}$ that packs and encrypts $\{X^{a_j}\}_{k_1 \in [r]}$ where $j = k_0 + k_1 \cdot r^{\rho-\rho'}$ and the error of $\boldsymbol{d}'_{r-1}$ is sub-Gaussian with parameter $E_{\mathsf{mul}} = O(u^3 r \sqrt{\tilde{N}\ell} E_D) + O(u^2 r E'_0)$ based on Lemma 6. After the external products, based on Lemma 7, unpack $\boldsymbol{d}'_{r-1}$ to get an REG ciphertext of X^{a_j} with an error sub-Gaussian with parameter $E_{\mathsf{unp}} = E_{\mathsf{mul}} + O(uE_{\mathsf{ks}}\sqrt{\tilde{N}\ell}) = O(UAE_{\rho'+1}) + O(UE_{\rho'+1}) + O(B) \stackrel{\mathsf{def}}{=} E_{\rho'}$, where $U = u^2 r$, $A = u\|z\|_2 \sqrt{\tilde{N}\ell}$, $B = u^{4.5} r \|z\|_2 \tilde{N}\ell E_{\mathsf{ks}}$. In addition, due to our special design in the structure of $\mathsf{HomINTT}$, when $\boldsymbol{B}_{0,k_0}$ is an REG ciphertext of $vX^{b_{0,k_0}}$ with an error sub-Gaussian with parameter $\hat{E}_{\rho'+1}$, by the above operations, we can obtain an REG ciphertext of vX^{a_j} with an error sub-Gaussian with parameter $E_{\mathsf{unp}} = O(UAE_{\rho'+1}) + O(U\hat{E}_{\rho'+1}) + O(B) \stackrel{\mathsf{def}}{=} \hat{E}_{\rho'}$, and v does not affect the error. Besides, when $\rho' = 1$, we extract an RLWE ciphertext $\boldsymbol{B}_{0,k_0,0}$ of $vX^{b_{0,k_0}}$ from $\boldsymbol{B}_{0,k_0}$ to construct $\boldsymbol{d}'_0$, resulting in less cost. After all iterations, $\mathsf{HomINTT}$ outputs RLWE ciphertexts of $\{vX^{\varphi_j}\}_{j \in [N]}$ with errors sub-Gaussian with parameter $\hat{E}_1 + O(\sqrt{q\ell} E_{\mathsf{ks}}) = O((UA)^\rho E_0) + O(U^\rho E_1) + O((UA)^{\rho-1} B)$ where $\sum_{j=0}^{N-1} \varphi_j X^j = \mathsf{INTT}((\tilde{\varphi}_i)_{i \in [N]})$. In the complexity analysis in Section 3.3, we will see that our homomorphic inverse NTT algorithm reduces the complexity from $O(N^{1+1/\rho})$ in [15] to $\tilde{O}(N)$. The analysis is summarized as Theorem 1.

Theorem 1. *Let $\boldsymbol{C}_i \in \mathsf{REG}^Q_{z,\mathcal{R}}(\mu_i)$ with an error sub-Gaussian with parameter B_i for $i \in [N]$ where $\mu_i = X^{\tilde{\varphi}_i}$ and $B_i = E_0$ for $i \in [N] \setminus \{0\}$, $\mu_0 = vX^{\tilde{\varphi}_0}$, $B_0 = E_1$ and $v \in \mathcal{R}$. If $\{\boldsymbol{c}'_j\}_{j \in [N]} \leftarrow \mathsf{HomINTT}(\{\boldsymbol{C}_j\}_{j \in [N]}, 1, \rho, \boldsymbol{AK}, \overline{\boldsymbol{EK}}, \overline{\boldsymbol{TK}}, \boldsymbol{SwK})$, then for $j \in [N]$, we have $\boldsymbol{c}'_j \in \mathsf{RLWE}^{Q,1}_{z,\mathcal{R}}(vX^{\varphi_j})$ with an error sub-Gaussian with parameter $E_{\mathsf{intt}} = O((UA)^\rho E_0) + O(U^\rho E_1) + O((UA)^{\rho-1} B)$, $U = u^2 r$, $A = u\|z\|_2 \sqrt{\tilde{N}\ell}$, $B = u^{4.5} r \|z\|_2 \tilde{N}\ell E_{\mathsf{ks}}$, and $\sum_{j=0}^{N-1} \varphi_j X^j = \mathsf{INTT}((\tilde{\varphi}_i)_{i \in [N]})$.*

3.3 Amortized Functional Bootstrapping

For any function f, given ciphertexts $\{\boldsymbol{c}_i\}_{i \in [N]}$ and an REG ciphertext $\boldsymbol{V}$ that encodes and encrypts f, where $\boldsymbol{c}_i$ is an LWE ciphertext of μ_i under the secret key

$\boldsymbol{s}_{\mathsf{in}} = (s_{\mathsf{in},i})_{i\in[n]}$ for $i \in [N]$, our amortized functional bootstrapping algorithm $\mathsf{BatchFB}$ computes LWE ciphertexts of $\{f(\mu_i)\}_{i\in[N]}$ with low noise.

The keys used in $\mathsf{BatchFB}$ are as follows. Let $\boldsymbol{SwK} \in \mathsf{REG}^{Q}_{z,\tilde{\mathcal{R}}}(z^2)$, $\boldsymbol{AK} = \{\boldsymbol{AK}_{\psi_k} \in \mathsf{REG}^{Q}_{z,\mathcal{R}}(\psi_k(z))\}_{k\in\mathbb{Z}_q^*}$, $\boldsymbol{PaK} = (\boldsymbol{Pak}_i \in \mathsf{REG}^{q}_{s,\hat{\mathcal{R}}}(s_{\mathsf{in},i}))_{i\in[n]}$, $\boldsymbol{BK} = \{\boldsymbol{BK}_i\}_{i\in[N]}$, $\overline{\boldsymbol{TK}} = \{\boldsymbol{TK}, \boldsymbol{TK}'\}$, $\overline{\boldsymbol{EK}} = \{\boldsymbol{EK}, \boldsymbol{EK}'\}$, where $s \in \hat{\mathcal{R}}_q$, $\tilde{\boldsymbol{s}} = \mathsf{NTT}(s)$, $\boldsymbol{BK}_0 \in \mathsf{RGSW}^{Q}_{z,\mathcal{R}}(X^{\tilde{s}_0})$, $\boldsymbol{BK}_i \in \mathsf{REG}^{Q}_{z,\mathcal{R}}(X^{\tilde{s}_i})$ for $i \in [N]\setminus\{0\}$, $\boldsymbol{EK} = \{\boldsymbol{EK}_b \in \mathsf{REG}^{Q}_{z,\tilde{\mathcal{R}}}(zq_{1-b}'^{-1})\}_{b\in[2]}$, $\boldsymbol{EK}' = \{\boldsymbol{EK}'_b \in \mathsf{REG}^{Q}_{z,\tilde{\mathcal{R}}_2}(zq_b'^{-1}q_2'^{-1})\}_{b\in[2]}$, $\boldsymbol{TK} = \{\boldsymbol{TK}_\theta \in \mathsf{REG}^{Q}_{z,\tilde{\mathcal{R}}}(\theta(z))\}_{\theta\in L}$, $L = \cup_{i=0}^{1}(\cup_{j=0}^{t_i-2}\mathsf{Gal}(T_{i,j}/T_{i,j+1}))$, $\boldsymbol{TK}' = \{\boldsymbol{TK}'_\theta \in \mathsf{REG}^{Q}_{z,\tilde{\mathcal{R}}_2}(\theta(z))\}_{\theta\in L'}$, $L' = \cup_{j=0}^{t_2-2}\mathsf{Gal}(T_{2,j}/T_{2,j+1})$. And let the errors of the keys be sub-Gaussian with parameter E_{ks}.

Our method (see Algorithm 3) consists of the following steps. Given the above LWE ciphertexts $\{\boldsymbol{c}_i\}_{i\in[N]}$, first pack them into a ciphertext $\boldsymbol{c} = (a, b) \in \mathsf{RLWE}^{q,\Delta}_{s,\hat{\mathcal{R}}}(\sum_{i=0}^{N-1}\mu_i X^i)$, and then apply NTT on a and $b + \lfloor\frac{q}{4}\rfloor$, obtaining $\tilde{\boldsymbol{a}} = (\tilde{a}_i)_{i\in[N]}$ and $\tilde{\boldsymbol{b}} = (\tilde{b}_i)_{i\in[N]}$ respectively. Then, for each $i \in \{1, \ldots, N-1\}$, perform blind rotation on the key $\boldsymbol{BK}_i$ by an automorphism and a polynomial multiplication, producing an ciphertext $\boldsymbol{C}_{\mathsf{br},i} \in \mathsf{REG}^{Q}_{z,\mathcal{R}}(X^{\tilde{a}_i\cdot\tilde{s}_i-\tilde{b}_i})$ with an error sub-Gaussian with parameter $O(\sqrt{q\ell}E_{\mathsf{ks}})$ based on Lemma 2, where we assume $\tilde{a}_i \neq 0$ for $i \in [N]$. Particularly, we use $\boldsymbol{V}$ to update $\boldsymbol{BK}_0$ to be an REG ciphertext of $v(X^{\tilde{a}_0^{-1}})\cdot X^{\tilde{s}_i}$ by an automorphism and an external product, where $v = v(X) = \Delta'\sum_{i=0}^{q-2}\sum_{j=0}^{i} f(\lfloor\frac{j-\lfloor\frac{q}{4}\rfloor}{\Delta}\rceil)X^i$ is the plaintext of $\boldsymbol{V}$ for $\Delta' \in \mathbb{Z}^+$, and $\boldsymbol{V}$ has an error sub-Gaussian with parameter E_{tv}. Then, by the blind rotation, we get $\boldsymbol{C}_{\mathsf{br},0} \in \mathsf{REG}^{Q}_{z,\mathcal{R}}(vX^{\tilde{a}_0\cdot\tilde{s}_0-\tilde{b}_0})$ whose error is sub-Gaussian with parameter $E_{\mathsf{tv}} + O(\sqrt{q\ell}E_{\mathsf{ks}})$ based on Lemma 1 and Lemma 2. Next, based on Theorem 1, execute our improved homomorphic inverse NTT algorithm on $\{\boldsymbol{C}_{\mathsf{br},i}\}_{i\in[N]}$ to obtain ciphertexts $\{\boldsymbol{c}_{\mathsf{intt},i}\}_{i\in[N]}$ where $\boldsymbol{c}_{\mathsf{intt},i} \in \mathsf{RLWE}^{Q,1}_{z,\mathcal{R}}(vX^{\varphi_i})$ and its error is sub-Gaussian with parameter $E_{\mathsf{intt}} = O(u^{2\rho}r^{\rho}E_{\mathsf{tv}}) + O(r^{\rho}u^{3\rho+1.5}\|z\|_2^{\rho}(\tilde{N}\ell)^{\frac{\rho+1}{2}}E_{\mathsf{ks}})$ for $i \in [N]$ and $\sum_{i=0}^{N-1}\varphi_i X^i = -(b - a\cdot s + \lfloor\frac{q}{4}\rfloor) \pmod q$. Lastly, extract the LWE ciphertext $\boldsymbol{c}_{\mathsf{out},i}$ of $f(\mu_i)$ from $\boldsymbol{c}_{\mathsf{intt},i}$ for each $i \in [N]$ and then output $\{\boldsymbol{c}_{\mathsf{out},i}\}_{i\in[N]}$ where $\boldsymbol{c}_{\mathsf{out},i} \in \mathsf{LWE}^{Q,\Delta'}_{\boldsymbol{z}}(f(\mu_i))$ and its error is sub-Gaussian with parameter $E_{\mathsf{out}} = E_{\mathsf{intt}}$ for $i \in [N]$ based on Lemma 4. Note that our amortized functional bootstrapping algorithm pulls in the test polynomial v before the blind rotation rather than after the homomorphic inverse NTT, allowing lower noise and higher precision in calculating any functions, compared with similar works [26,15,27,20]. The above analysis is summarized as Theorem 2.

Theorem 2. *Let $\boldsymbol{c}_i \in \mathsf{LWE}^{q,\Delta}_{\boldsymbol{s}_{\mathsf{in}}}(\mu_i)$ for $i \in [N]$. Let $\boldsymbol{V} \in \mathsf{REG}^{Q}_{z,\mathcal{R}}(v)$ with an error sub-Gaussian with parameter $E_{\mathsf{tv}} \approx E_{\mathsf{ks}}$ where $v = \Delta'\sum_{i=0}^{q-2}\sum_{j=0}^{i} f(\lfloor\frac{j-\lfloor\frac{q}{4}\rfloor}{\Delta}\rceil)X^i$ and E_{tv} is independent of f for $\Delta' \in \mathbb{Z}^+$ and any function $f : \mathbb{Z}_q \to \mathbb{Z}_Q$. Let $\boldsymbol{c} \leftarrow \mathsf{PackLWE}(\{\boldsymbol{c}_i\}_{i\in[N]}, \boldsymbol{PaK})$ such that $[\langle\boldsymbol{c}, (-s, 1)\rangle]_q = \Delta\mu + e$ where $\mu = \sum_{i=0}^{N-1}\mu_i X^i$ and e is an error. If $\|\Delta\mu + e\| \leq \lfloor\frac{q}{4}\rfloor$ and $\|e\| < \frac{\Delta}{2}$, then $\mathsf{BatchFB}(\{\boldsymbol{c}_i\}_{i\in[N]}, \boldsymbol{BK}, \boldsymbol{AK}, \overline{\boldsymbol{EK}}, \overline{\boldsymbol{TK}}, \boldsymbol{SwK}, \boldsymbol{PaK}, \boldsymbol{V}, \rho, q, Q)$ outputs LWE*

Algorithm 3 BatchFB

Input: LWE ciphertexts $\{\boldsymbol{c}_i\}_{i\in[N]}$, $\boldsymbol{BK}, \boldsymbol{AK}, \overline{\boldsymbol{EK}}, \overline{\boldsymbol{TK}}, \boldsymbol{SwK}, \boldsymbol{PaK}, \boldsymbol{V}, \rho, q, Q$
Output: LWE ciphertexts $\{\boldsymbol{c}_{\mathsf{out},i}\}_{i\in[N]}$
1: $\boldsymbol{c} = (a, b) \leftarrow \mathsf{PackLWE}(\{\boldsymbol{c}_i\}_{i\in[N]}, \boldsymbol{PaK})$
2: $\tilde{\boldsymbol{a}} = (\tilde{a}_i)_{i\in[N]} = \mathsf{NTT}(a), \tilde{\boldsymbol{b}} = (\tilde{b}_i)_{i\in[N]} = \mathsf{NTT}(b + \lfloor \frac{q}{4} \rfloor)$
3: $\boldsymbol{BK}_0 \leftarrow \boldsymbol{BK}_0 \boxtimes \mathsf{Aut}(\boldsymbol{V}, \theta, \boldsymbol{AK}_\theta)$ where $\theta = \psi_{[\tilde{a}_0^{-1}]_q}$
4: **for** $i = 0$ to $N-1$ **do**
5: $\quad \boldsymbol{C}_{\mathsf{br},i} \leftarrow X^{-\tilde{b}_i} \cdot \mathsf{Aut}(\boldsymbol{BK}_i, \theta_i, \boldsymbol{AK}_{\theta_i})$ where $\theta_i = \psi_{\tilde{a}_i}$
6: $\{\boldsymbol{c}_{\mathsf{intt},i}\}_{i\in[N]} \leftarrow \mathsf{HomINTT}(\{\boldsymbol{C}_{\mathsf{br},i}\}_{i\in[N]}, 1, \rho, \boldsymbol{AK}, \overline{\boldsymbol{EK}}, \overline{\boldsymbol{TK}}, \boldsymbol{SwK})$
7: **for** $i = 0$ to $N-1$ **do**
8: $\quad \boldsymbol{c}_{\mathsf{out},i} \leftarrow \mathsf{Extract}(\boldsymbol{c}_{\mathsf{intt},i}, Q)$

Table 1. Number of $\odot$ in basic algorithms and comparison of different functional bootstrapping algorithms

Algorithm	$\boxtimes$	Aut	Swicth	Trace	BatchMul	BatchAut	UnPack
Cost	2	1	ℓ	$O(\log u)$	$O(\log u)$	$O(\log u)$	$O(r \log u)$
Algorithm	BatchFB	[10,7,18]	[21,25]	[26]	[15,27]	[19]	[20]
Amortized cost	$\tilde{O}(1)$	$O(n)$	$O(n)$	$\tilde{O}(3^\rho \cdot n^{1/\rho})$	$O(\rho \cdot n^{1/\rho})$	$\tilde{O}(n^{0.75})$	$\tilde{O}(1)$
Multi-bit plaintext	✓	×	✓	×	✓	×	×
Encrypted function	✓	×	✓	×	×	×	×

ciphertexts $\{\boldsymbol{c}_{\mathsf{out},i} \in \mathsf{LWE}_{\boldsymbol{z}}^{Q,\Delta'}(f(\mu_i))\}_{i\in[N]}$ *with errors sub-Gaussian with parameter* $E_{\mathsf{out}} = O(u^2 \rho r^\rho E_{\mathsf{tv}}) + O(r^\rho u^{3\rho+1.5} \|z\|_2^\rho (\tilde{N}\ell)^{\frac{\rho+1}{2}} E_{\mathsf{ks}})$.

The computational cost of Algorithm 3 is dominated by blind rotation and homomorphic inverse NTT. In this paper, we measure the complexity by the number of $\odot$ operations, and the complexity of each basic algorithm is presented in Table 1. Then, in blind rotation, the complexity is $O(N)$. Denote $T(N)$ as the complexity of HomINTT for N inputs. And we observe that there are r calls of BatchAut, Swicth and BatchMul, and one call of UnPack for each $k_0 \in [r^{\rho-\rho'}]$ in the ρ'-th layer of HomINTT ($\rho' \in \{1, \ldots, \rho\}$). Besides, one operation on an REG ciphertext contains ℓ same operations on RLWE ciphertexts. Similar to [20], determine all the parameters in terms of the security parameter λ, and set $n = O(\lambda)$, $q = \tilde{O}(\sqrt{n})$, $N = O(n)$, $\ell = \tilde{O}(1)$, $u \approx O(\lambda^{1/4-o(1)})$, $r = O(\lambda^{0.2})$, $\rho = 5$. Then $T(N) = rT(r^{\rho-1}) + r^{\rho-1} \cdot (r \cdot (O(\log u) + \ell + O(\log u)) + O(r \log u)) + N = rT(r^{\rho-1}) + r^{\rho-1} \cdot \tilde{O}(r) + N = rT(r^{\rho-1}) + 2 \cdot \tilde{O}(N) = r(rT(r^{\rho-2}) + r^{\rho-2} \cdot \ell \cdot \tilde{O}(r)) + 2 \cdot \tilde{O}(N) = r^2 T(r^{\rho-2}) + 3 \cdot \tilde{O}(N) = \cdots = r^{\rho-1} \cdot \tilde{O}(r) + \tilde{O}(\rho N) = \tilde{O}(N)$. So the amortized complexity of Algorithm 3 is $\tilde{O}(1)$ per refreshed message.

Compared with previous works, our amortized functional bootstrapping algorithm has many advantages. As shown in Table 1, the amortized complexities of the algorithm of [20] and our algorithm are the best. In addition, our algorithm supports homomorphic evaluation of any functions on multi-bit messages, similar to [21,25,15,22]. But the algorithms of [10,7,18,26,27,19,20] only support Boolean functions on single-bit messages. Furthermore, our method supports homomorphic evaluation of an encrypted function f, but the methods

of [10,26,15,27,19,20,22] can not. This property is very useful in applications that require protecting the privacy of computational models, such as privacy-preserving machine learning against model extraction attacks.

For the evaluation of a public Boolean function, the output errors of the amortized bootstrapping algorithm of [20] are sub-Gaussian with parameter $E'_{\mathsf{out}} = O(\sqrt{N}(\log q)r^{\rho}u^{3\rho+3}\|z\|^{\rho+1}(\tilde{N}\ell)^{\rho+\frac{1}{2}}E_{\mathsf{ks}}) \approx \tilde{O}(N^{11.5}E_{\mathsf{ks}})$. But according to Theorem 2, the sub-Gaussian parameters of our output errors are $E_{\mathsf{out}} \approx \tilde{O}(N^{9.375}E_{\mathsf{ks}})$ and thus are reduced by at least a factor of $\tilde{O}(N^{2.125})$. For any public function f, the amortized complexity of the BFV-based functional bootstrapping algorithm of [22] is also $\tilde{O}(1)$. In [22], the multiplicative depth of the bootstrapping is $3 + \log q$, and thus the sub-Gaussian parameters of the output errors are about $O(B_f(qN\|z\|)^{3+\log q}E_{\mathsf{ks}})$, where B_f is the upper bound of $|f(x)|$, $q = 2N$ and $\log q \geq 16$, based on the error growth of the BFV multiplication in [11]. In our algorithm, the output errors are independent of f and their sub-Gaussian parameters are reduced by at least a factor of $\tilde{O}(B_f N^{28.625})$ compared with [22]. So, for correctness, the modulus of bootstrapping keys in our algorithm is smaller than that of [20] and much smaller than that of [22].

4 Conclusion

We extend the amortized homomorphic automorphism algorithm of [20] to perform different automorphisms and key switching on multiple ciphertexts simultaneously. Next, we improve the homomorphic inverse NTT algorithm of [15] to reduce costs and then use it to construct a new amortized functional bootstrapping algorithm whose amortized complexity is $\tilde{O}(1)$ to support the evaluation of any public or encrypted function f on multi-bit messages. Additionally, the output errors of our bootstrapping algorithm are sub-Gaussian with parameter $\tilde{O}(EN^{9.375})$, independent of f, where N is the number of input ciphertexts and E is the sub-Gaussian parameter of bootstrapping keys. In particular, for any public function f, the sub-Gaussian parameters of the output errors are reduced by a factor of $\tilde{O}(|f(x)|N^{28.625})$ compared with [22]. For applications, our bootstrapping algorithm can be used in scenarios where the privacy of computational models and data needs to be protected. In our future work, we intend to find ways to further optimize and implement our bootstrapping algorithm.

Acknowledgments. Li-Ping Wang is supported by the National Natural Science Foundation of China under Grant No. 62372446. The research of Huaxiong Wang is supported by Singapore Ministry of Education Academic Research Fund Tier 2 Grant T2EP20223-0028.

References

1. Alperin-Sheriff, J., Peikert, C.: Faster bootstrapping with polynomial error. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 297–314. Springer, Heidelberg (2014), https://doi.org/10.1007/978-3-662-44371-2_17

2. Boura, C., Gama, N., Georgieva, M., Jetchev, D.: Simulating homomorphic evaluation of deep learning predictions. In: Dolev, S., Hendler, D., Lodha, S., Yung, M. (eds.) CSCML 2019. LNCS, vol. 11527, pp. 212–230. Springer, Cham (2019), https://doi.org/10.1007/978-3-030-20951-3_20
3. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical gapsvp. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 868–886. Springer, Heidelberg (2012), https://doi.org/10.1007/978-3-642-32009-5_50
4. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. ACM Transactions on Computation Theory **6**(3), 13:1–13:36 (2014)
5. Brakerski, Z., Langlois, A., Peikert, C., Regev, O., Stehlé, D.: Classical hardness of learning with errors. In: Proceedings of the 45th Annual ACM Symposium on Theory of Computing, STOC 2013. pp. 575–584. ACM (2013)
6. Carpov, S., Izabachène, M., Mollimard, V.: New techniques for multi-value input homomorphic evaluation and applications. In: Matsui, M. (ed.) CT-RSA 2019. LNCS, vol. 11405, pp. 106–126. Springer, Cham (2019), https://doi.org/10.1007/978-3-030-12612-4_6
7. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster packed homomorphic operations and efficient circuit bootstrapping for TFHE. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10624, pp. 377–408. Springer, Cham (2017), https://doi.org/10.1007/978-3-319-70694-8_14
8. Chillotti, I., Joye, M., Paillier, P.: Programmable bootstrapping enables efficient homomorphic inference of deep neural networks. In: Dolev, S., Margalit, O., Pinkas, B., Schwarzmann, A.A. (eds.) CSCML 2021. LNCS, vol. 12716, pp. 1–19. Springer, Cham (2021), https://doi.org/10.1007/978-3-030-78086-9_1
9. Chillotti, I., Ligier, D., Orfila, J., Tap, S.: Improved programmable bootstrapping with larger precision and efficient arithmetic circuits for TFHE. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021. LNCS, vol. 13092, pp. 670–699. Springer, Cham (2021), https://doi.org/10.1007/978-3-030-92078-4_23
10. Ducas, L., Micciancio, D.: FHEW: bootstrapping homomorphic encryption in less than a second. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 617–640. Springer, Heidelberg (2015), https://doi.org/10.1007/978-3-662-46800-5_24
11. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. IACR Cryptology ePrint Archive, p. 144 (2012)
12. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009. pp. 169–178. ACM (2009)
13. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 75–92. Springer, Heidelberg (2013), https://doi.org/10.1007/978-3-642-40041-4_5
14. Guimarães, A., Borin, E., Aranha, D.F.: Revisiting the functional bootstrap in TFHE. IACR Transactions on Cryptographic Hardware and Embedded Systems **2021**(2), 229–253 (2021)
15. Guimarães, A., Pereira, H.V.L., Leeuwen, B.V.: Amortized bootstrapping revisited: Simpler, asymptotically-faster, implemented. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023. LNCS, vol. 14443, pp. 3–35. Springer, Singapore (2023), https://doi.org/10.1007/978-981-99-8736-8_1

16. Joye, M., Walter, M.: Liberating TFHE: programmable bootstrapping with general quotient polynomials. In: Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography. pp. 1–11. ACM (2022)
17. Kluczniak, K., Schild, L.: FDFB: full domain functional bootstrapping towards practical fully homomorphic encryption. IACR Transactions on Cryptographic Hardware and Embedded Systems **2023**(1), 501–537 (2023)
18. Lee, Y., Micciancio, D., Kim, A., Choi, R., Deryabin, M., Eom, J., Yoo, D.: Efficient FHEW bootstrapping with small evaluation keys, and applications to threshold homomorphic encryption. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023. LNCS, vol. 14006, pp. 227–256. Springer, Cham (2023), https://doi.org/10.1007/978-3-031-30620-4_8
19. Liu, F., Wang, H.: Batch bootstrapping I: - A new framework for SIMD bootstrapping in polynomial modulus. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023. LNCS, vol. 14006, pp. 321–352. Springer, Cham (2023), https://doi.org/10.1007/978-3-031-30620-4_11
20. Liu, F., Wang, H.: Batch bootstrapping II: - bootstrapping in polynomial modulus only requires o(1) FHE multiplications in amortization. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023. LNCS, vol. 14006, pp. 353–384. Springer, Cham (2023), https://doi.org/10.1007/978-3-031-30620-4_12
21. Liu, Z., Micciancio, D., Polyakov, Y.: Large-precision homomorphic sign evaluation using FHEW/TFHE bootstrapping. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022. LNCS, vol. 13792, pp. 130–160. Springer, Cham (2022), https://doi.org/10.1007/978-3-031-22966-4_5
22. Liu, Z., Wang, Y.: Amortized functional bootstrapping in less than 7 ms, with $\tilde{o}(1)$ polynomial multiplications. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023. LNCS, vol. 14443, pp. 101–132. Springer, Singapore (2023), https://doi.org/10.1007/978-981-99-8736-8_4
23. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 1–23. Springer, Heidelberg (2010), https://doi.org/10.1007/978-3-642-13190-5_1
24. Lyubashevsky, V., Peikert, C., Regev, O.: A toolkit for ring-lwe cryptography. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 35–54. Springer, Heidelberg (2013), https://doi.org/10.1007/978-3-642-38348-9_3
25. Ma, S., Huang, T., Wang, A., Zhou, Q., Wang, X.: Fast and accurate: Efficient full-domain functional bootstrap and digit decomposition for homomorphic computation. IACR Transactions on Cryptographic Hardware and Embedded Systems **2024**(1), 592–616 (2024)
26. Micciancio, D., Sorrell, J.: Ring packing and amortized FHEW bootstrapping. In: 45th International Colloquium on Automata, Languages, and Programming, ICALP 2018. LIPIcs, vol. 107, pp. 100:1–100:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2018)
27. Micheli, G.D., Kim, D., Micciancio, D., Suhl, A.: Faster amortized FHEW bootstrapping using ring automorphisms. IACR Cryptology ePrint Archive, p. 112 (2023)
28. Peikert, C., Regev, O., Stephens-Davidowitz, N.: Pseudorandomness of ring-lwe for any ring and modulus. In: Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017. pp. 461–473. ACM (2017)
29. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Proceedings of the 37th Annual ACM Symposium on Theory of Computing, STOC 2005. pp. 84–93. ACM (2005)