CKKS-Based Homomorphic Encryption Architecture using Parallel NTT Multiplier

Tuy Tan Nguyen
School of Informatics, Computing, and Cyber Systems
Northern Arizona University
Flagstaff, AZ 86011, USA
tuy.nguyen@nau.edu

Jisu Kim, Hanho Lee
Department of Information and Communication Engineering
Inha University
Incheon 22212, Korea
hhlee@inha.ac.kr

2023 IEEE International Symposium on Circuits and Systems (ISCAS) | 978-1-6654-5109-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/ISCAS46773.2023.10181714

***Abstract*—This paper presents a high-throughput CKKS-based encryption architecture for homomorphic encryption. By deploying a parallel number theoretic transform (NTT) multiplier architecture, the polynomial multiplication is significantly accelerated. Additionally, the modular multiplier is also improved by efficiently implementing using digital signal processing resources. The proposed NTT multiplier and homomorphic encryption architecture are evaluated using Xilinx Vivado and Xilinx XCU250 FPGA board. The evaluation results demonstrate that the proposed NTT multiplier helps improve the throughput of polynomial multiplication by at least $1.5\times$ compared to the most recent works. The efficiency of the proposal NTT multiplier, calculated by throughput per LUT or Slice, is much better than that of existing studies. The proposed homomorphic encryption architecture using the proposed NTT multiplier offers a high throughput of 32.7 Gbps.**

***Index Terms*—CKKS, homomorphic encryption, learning with errors, number theoretic transform**

I. Introduction

Cloud-based computing services offer considerable benefits for low-cost computation. However, it also introduces security and privacy concerns where breaching user data become a top threat. In that scene, homomorphic encryption (HE) has been considered a perfect candidate to prevent data leakage. HE enables calculating on fully encrypted data and helps protect users' data from potential attacks. In HE, the client's data are encrypted before sending to the cloud server to perform computation. The cloud server computes the encrypted data without caring about the original data. The resultant computation is also in the encrypted form, which only the client holding the secret key can decrypt. Therefore, client's data are completely protected.

In 1979, Rivest, Adleman, and Dertouzos introduced a concept of computation on encrypted data [1]. After that, there are several HE schemes presented in the literature. Somewhat homomorphic encryption (SHE), which can only handle a limited class of uncomplicated functions, was introduced in [2], [3]. Gentry [4] proved that SHE can be elevated to fully HE (FHE) by introducing a bootstrapping technique. However, this technique was extremely costly and inefficient [5]. In [6], Gentry and Halevi presented the practical FHE implementation to efficiently solve the infamous bottleneck of FHE schemes. Over the last decade, improving performance of HE has become an emerging topic attracted a great attention from research community. Studies on SHE and FHE can be found in [3], [7]–[15]. Practical HE schemes proposed in [9] are based on learning with errors (LWE) problem and ring LWE (RLWE). The details of LWE and RLWE problems can be found in [9], [16]. Authors in [12] proposed a HE scheme named CKKS with a capability of approximate arithmetic on real or complex numbers. Although the CKKS-based HE can support the truncation of encrypted values and batch computation, it introduces a massive latency to computation time. Among the operations that contribute to the delay of the CKKS scheme, polynomial multiplication is a majority.

In this paper, we propose a novel NTT polynomial multiplier to accelerate the polynomial multiplication in the CKKS-based HE scheme. The proposed NTT multiplier is then deployed into the HE architecture to facilitate the encryption operation. Furthermore, an efficient architecture of the modulo multiplier is introduced to speed up the modulo operation. As a result, the proposed parallel NTT multiplier and the CKKS-based HE architecture obtain high throughputs of 33.5 Gbps and 32.7 Gbps, respectively.

The remaining of this paper is structured as follows. Section II briefly describes the CKKS-based HE algorithm. Section III presents the proposed parallel NTT multiplier and the CKKS-based HE architecture using the proposed multiplier. Section IV analyzes the performance of the proposed architecture. Finally, conclusions are given in Section V.

II. CKKS-Based Homomorphic Encryption Algorithm

In [12], Cheon et al. proposed an HE scheme named CKKS for the arithmetic of approximate numbers. In this algorithm, a base p and a modulus q_0 are fixed, and $q_l = p^l \cdot q_0$, where $0 < l \leq L$. For a security parameter λ, select a parameter $M = M(\lambda, q_L)$ for cyclotomic polynomial. A ciphertext of a level l is a vector in $R_{q_l}^k$, where $0 \leq l \leq L$, for a fixed k. CKKS scheme consists of five algorithms, KeyGen(), Enc(), Dec(), Add(), and Mult(), described as follows:

This work was supported by Northern Arizona University, USA, and in part by the KEIT grant funded by the Ministry of Trade, Industry Energy (MOTIE, Korea) (No. 20010589), and in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2021R1A2C1011232).

- KeyGen(1^λ): Generate a secret key sk, a public key pk, and an evaluation key evk.
- Enc$_{pk}(m)$: Construct a ciphertext $c \in R_{q_l}^k$ for a given polynomial $m \in R$. Encryption satisfies $(c, sk) = m + e \pmod{q_L}$ for some small e.
- Dec$_{sk}(c)$: From a layer L ciphertext c and a secret key sk, produce a polynomial $m' \leftarrow (c, sk) \bmod q_l$.

The HE algorithms satisfy the following properties:

- Add(c_1, c_2): Output an encryption $c_1 + c_2$, where c_1 and c_2 are the ciphertexts of input polynomials m_1 and m_2, respectively. The error of $c_1 + c_2$ is bound by sum of two errors in input ciphertexts c_1 and c_2.
- Mult$_{evk}(c_1, c_2)$: Output of the multiplication between the two ciphertexts c_1 and c_2 satisfies $(c_{mult}, sk) = (c_1, sk) \cdot (c_2, sk) + e_{mult} \bmod q_l$.

CKKS algorithm requires a rescaling procedure $R_{S_{l \to l'}}(c)$ on the ciphertex c [12].

III. Proposed CKKS-Based Homomorphic Encryption Architecture

A. Proposed NTT Architecture

Polynomial multiplication is the most complex arithmetic operation and the major performance bottleneck in homomorphic computations. In this paper, we design a parallel NTT multiplier with specific parameters of the CKKS-based HE algorithm. Specifically, input data are divided into multiple paths to be processed simultaneously. In addition, the proposed NTT architecture is fully pipelined to operate at a high speed. As a result, the proposed NTT multiplier can accelerate the polynomial multiplication operation and offer a high throughput compared to existing architectures. Figure 1 shows a 2-parallel multi-path delay feedback-based NTT architecture with $n = 2^{14}$. Input data are divided into two paths with odd and even indexes, a_{2i} and a_{2i+1}. For $n = 2^{14}$, the proposed NTT architecture consists of 14 stages. The first 13 stages of NTT architecture use the processing element 1 (PE1), and the last stage of NTT architecture uses the processing element 2 (PE2). The design of processing elements PE1 and PE2 is illustrated in Figure 1. In the design of PE1, modular multiplier block (MM) performs multiplication between input data and the twiddle factor (TF) value ω^i loaded from Block RAMs (BRAMs). PE2 is used at the last stage of NTT and performs the butterfly operation on two inputs. The design of the inverse NTT (INTT) is similar to the NTT design by placing the PE2 at the first stage.

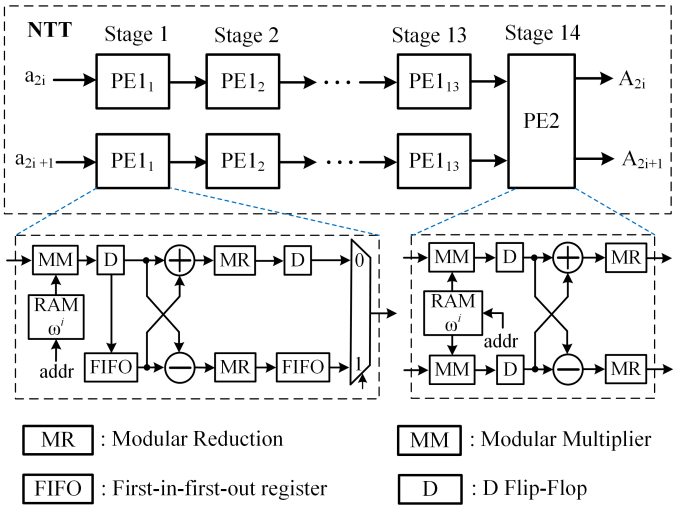

Fig. 1. Proposed NTT Architecture.

B. Proposed Modulo Multiplier Architecture

Modular multiplier performs multiplication between two inputs and then modulo by a number q (e.g., $C = A \cdot B \bmod q$). In this paper, we modify the DSP-based fully pipelined IntMult design presented in [18] for 60-bit Barrett modulo multiplier [17]. The proposed 60-bit Barrett modulo multiplier architecture is presented in Figure 2. The modulo multiplier consists of one IntMult and two HalfMult (HalfMult_T and HalfMult_q). The 60-bit Barrett ModMult needs a single 60-bit Full-IntMult and two 61-bit Half-IntMults. To efficiently use DSP48E2 slices for the 60-bit IntMult, the first operand is divided into two 26-bit operands and one 8-bit operand. The second operand is divided into three 17-bit operands and one a single 9-bit operand. To perform the partial multiplications in parallel, 12 DSP slices are needed. Figure 3 illustrates the processing order of 60-bit IntMult using DSP48E2 slice. In Figure 3, A and B are 60-bit input operands which are divided into partial operands A_i and B_j where $i = 0, 1, 2$ and $j = 0, 1, 2, 3$. A single DSP slice performs a partial multiplication for each stage (Stage 1: A_0B_0, Stage 2: A_0B_1, ..., Stage 12: A_2B_3).

The architectures of auxiliary multipliers HalfMult_T and HalfMult_q are shown in Figure 2. HalfMult_T multiplies the upper bits of the output of the IntMult. After that, the HalfMult_q calculates the lower 60 bits of the multiplication result from the IntMult. The HalfMult_q outputs only the lower 60-bits of the multiplication result, it therefore decreases

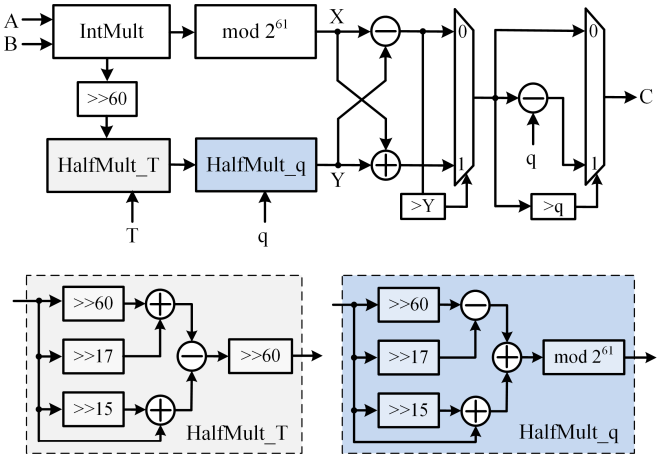

Fig. 2. Proposed Barrett-based Modulo Multiplier Architecture.

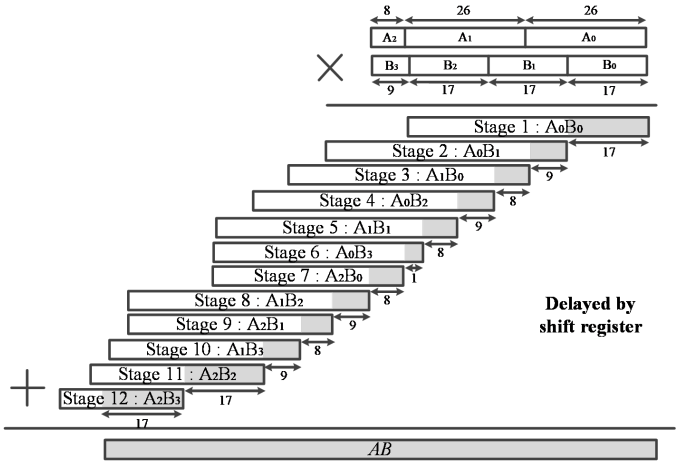

Fig. 3. DSP48E2 Resource Allocation for 60-bit Integer Multiplier.

the number of pipeline stages in the HalfMult_q. The output from HalfMult_q is then added with X or subtracted from X before comparing with Y and q to get the final output, as described in Figure 3.

C. CKKS-Based Homomorphic Encryption Architecture

In this paper, we introduce a CKKS-based homomorphic encryption architecture using the proposed parallel NTT multiplier and modulo multiplier to obtain a high throughput value. The proposed CKKS-based HE architecture is presented in Figure 4. The key generation process generates a public key that can be used in the encryption module with the input of 1^{λ}. The value of λ corresponds to the security parameter set by the homomorphic encryption standardization. Three error polynomials e_0, e_1, and Z_{op} are generated from the Gaussian sampler. Three error polynomials e_0, e_1, and Z_{op} are transformed using the proposed NTT architecture to get the value of $\hat{e}_0$, $\hat{e}_0$, $\hat{Z}_{op}$, respectively. Then $\hat{Z}_{op}$ is multiplied by $pk = (a, b)$ using a modular multiplier, and added to $\hat{e}_0$ and $\hat{e}_1$ to output results $aZ_{op}+e_0$ and $bZ_{op}+e_1$, as described in Figure 4. After passing through the ModSwitch module, the ciphertext c_0 is constructed by adding $bZ_{op}+e_1$ and the encoded message $m(X)$ through an encoder. Ciphertext c_1 is the result of ModSwitch with the input $aZ_{op} + e_0$. Figure 5 shows the CKKS-based homomorphic encryption architecture in detail. NTT operations are performed for parameters q_0, q_1, q_2, q_3, and P, corresponding to security level 5 as suggested by SEAL [19]. Thereafter, data corresponding to the last specific parameter P is reduced after passing through the modular switch module. The data size of the ciphertext is reduced by logP. In the encryption architecture structure, before the modular switch, 264-bit data performs key-size operations and after the ModSwitch, 204-bit data-size operations are performed.

IV. Implementation Results and Comparison

The proposed parallel NTT multiplier-based CKKS HE architecture is evaluated using Xilinx Vivado and Xilinx

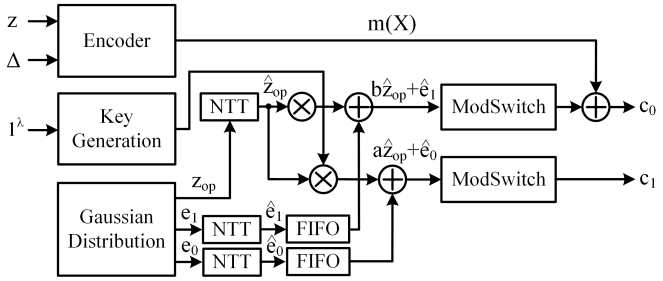

Fig. 4. Proposed CKKS-based Homomorphic Encryption Architecture.

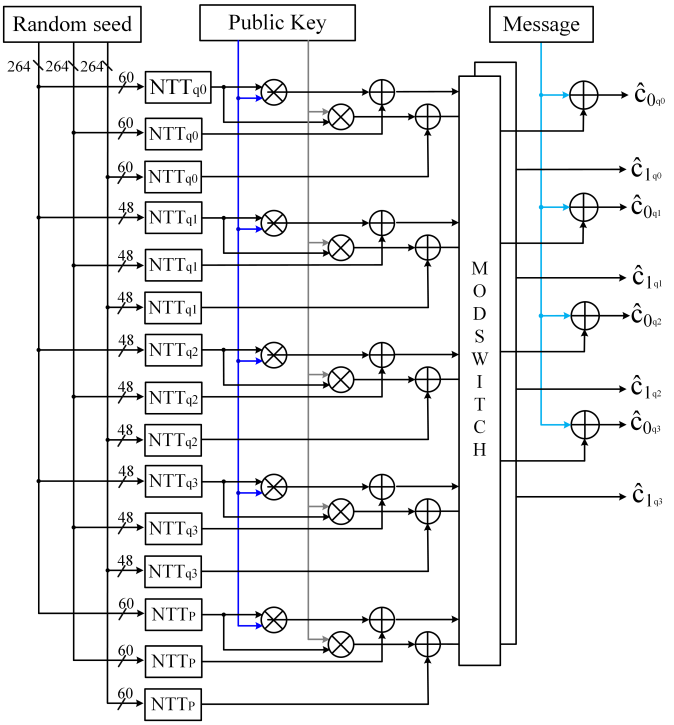

Fig. 5. Detailed CKKS-based Homomorphic Encryption Architecture.

XCU250 FPGA board. Verilog HDL is used as a modeling language. The implementation results are presented in Table I and Table II. The reported values of our design are for the parameter set (n, q) with $n = 2^{14}$ and $log_2 q = 60$.

Table I compares the hardware design of different NTT architectures. As can be seen from Table I, the proposed NTT architecture requires only 36.4K LUTs and 3.6K Slices, which are much smaller than that in [20]–[22]. The proposed NTT architecture also requires fewer BRAM and DSP resources than other works. Furthermore, the proposed NTT architecture obtains a high throughput of 33.5 Gbps thanks to the parallel operations. Since architectures in Table I are implemented on different devices, we use the normalized throughput per hardware unit (e.g., throughput/LUT, throughput/slice) as an additional parameter for a fair comparison. The results in Table I prove that our NTT architecture offers a better throughput per hardware unit compared to existing studies.

Table II shows the results of integrating the proposed NTT

TABLE I
COMPARISON OF DIFFERENT NTT HARDWARE ARCHITECTURES

	[20]	[22]	[21]	**This work**
Device	VX485T	VU190	VX690T	XCU250
n	2^{12}	2^{17}	2^{15}	2^{14}
$\log_2 q$	32	62	32	60
LUT	39.6K	365K	219.1K	36.4K
Slice	–	335K	90.7K	3.6K
BRAM	96	10163	193	60
DSP	224	1332	768	336
Freq. (MHz)	290	150	250	200
Latency (µs)	5.8	3280	50.9	29.3
Throu. (Gbps)	22.6	2.5	20.6	33.5
Norm. Throu.	1.1	0.12	**1**	**1.63**
Norm. Throu./LUT	6.1	0.07	**1**	**9.8**
Norm. Throu./Slice	–	0.03	**1**	**41**

TABLE II
CKKS-BASED HE ARCHITECTURE USING THE PROPOSED NTT MULTIPLIER ON XILINX XCU250 FPGA BOARD

LUT	**FF**	**BRAM**	**DSP**	**Freq.** (MHz)	**Latency** (µs)	**Throu.** (Gbps)
883K (51%)	897K (26%)	1563 (58%)	6042 (49%)	250	102.1	32.7

design into the CKKS-based HE architecture. As can be seen, the proposed CKKS-based HE architecture uses 883K LUTs, 897K Slices, 1563 BRAM, and 6042 DSPs, which are 51%, 26%, 58%, and 49% of the existing resources of XCU250 FPGA board. Therefore, the proposed HE architecture can be successfully implemented on the available FPGA board. Additionally, the proposed HE architecture can work at the highest frequency of 250 MHz, complete an HE operation in about 102 µs, and offer a high throughput of 32.7 Gbps.

V. CONCLUSION

This paper presents a high-throughput hardware architecture for CKKS-based homomorphic encryption. By deploying a parallel NTT multiplier architecture, the polynomial multiplication is significantly improved in terms of both latency and throughput. Furthermore, a new architecture is proposed to perform the Barrett modular multiplier in such a way to efficiently use the hardware resources of the FPGA board. As a result, the proposed CKKS-based HE architecture is successfully implemented on Xilinx XCU250 FPGA board to offer a high throughput of 32.7 Gpbs. Performance of the proposed NTT architecture is also better than existing works. Therefore, the proposed CKKS-based HE architecture can be applied in cloud service applications to offer high throughput.

REFERENCES

[1] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," *Found. Secure Comput.*, vol. 4, no. 11, pp. 169–180, 1978.

[2] A. Costache and N. P. Smart, "Which ring based somewhat homomorphic encryption scheme is best?," *Topics in Cryptology*, vol. 9610. Springer, Cham, Feb. 2016.

[3] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," *IACR Cryptology ePrint Archive*, vol. 2012, pp. 144, 2012.

[4] C. Gentry, "Fully homomorphic encryption using ideal lattices," *Annu. ACM Symp. Theory of Computing*, 2009, pp. 169–178.

[5] G. S. Çetin, E. Savaş and B. Sunar, "Homomorphic sorting with better scalability," *IEEE Trans. Parallel and Distributed Syst.*, vol. 32, no. 4, pp. 760–771, Apr. 2021.

[6] C. Gentry and S. Halevi, "Implementing Gentry's fully-homomorphic encryption scheme," *Annu. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2011, pp. 129–148.

[7] C. Gentry and S. Halevi, "Fully homomorphic encryption without squashing using depth-3 arithmetic circuits," *IACR Cryptol. ePrint Arch.*, vol. 2011, no. 279, 2011.

[8] Z. Brakerski, "Fully homomorphic encryption without modulus switching from classical GapSVP", *Annu. Cryptology Conf.*, Springer, Berlin, 2012, pp. 868–886.

[9] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "Fully homomorphic encryption without bootstrapping," *Electron. Colloq. Comput. Complexity*, 2011, Art. no. 111.

[10] A. López-Alt, E. Tromer, and V. Vaikuntanathan, "On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption", *44th Annu. ACM Symp. Theory of Computing (STOC)*, New York, USA, 2012, pp. 1219–1234.

[11] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds," *Advances in Cryptology - ASIACRYPT 2016*, vol. 10031, Springer, Berlin, Heidelberg, pp. 3–33, Dec. 2016.

[12] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," *Advances in Cryptology - ASIACRYPT, LNCS*, vol. 10031, Springer, Cham, Nov. 2017.

[13] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, "A full RNS variant of approximate homomorphic encryption," *Sel. Areas Cryptogr.*, vol. 11349, pp. 347–368, Jan. 2019.

[14] J. H. Cheon, A. Costache, R. C. Moreno, W. Dai, N. Gama, M. Georgieva, S. Halevi, M. Kim, S. Kim, K. Laine, and Y. Polyakov, "Introduction to homomorphic encryption and schemes," *Protecting Privacy Through Homomorphic Encryption 2021*, Springer, pp. 3–28, Jan. 2022.

[15] W. Jung, E. Lee, S. Kim, J. Kim, N. Kim, K. Lee, C. Min, J. H. Cheon, and J. Ahn, "Accelerating fully homomorphic encryption through architecture-centric analysis and optimization," *IEEE Access*, vol. 9, pp. 98772–98789, Jul. 2021.

[16] T. N. Tan and H. Lee, "High-secure low-latency ring-LWE cryptography scheme for biomedical images storing and transmitting," *2018 IEEE Int. Symp. Circuits Syst.* (ISCAS), Florence, Italy, 27–30 May 2018, pp. 1–4.

[17] P. Barrett, "Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor," *Advances in Cryptology*, 1987, pp.311–323.

[18] S. Kim, K. Lee, W. Cho, J. H. Cheon and R. A. Rutenbar, "FPGA-based accelerators of fully pipelined modular multipliers for homomorphic encryption," *2019 Int. Conf. ReConFigurable Computing and FPGAs (ReConFig)*, Cancun, Mexico, 09–11 Dec. 2019, pp. 1–8.

[19] Microsoft SEAL (release 3.6), https://github.com/Microsoft/SEAL, Microsoft Research, Redmond, WA, Nov. 2020.

[20] A. C. Mert, E. Ozturk, and E. Savas, "FPGA implementation of a run-time configurable NTT-based polynomial multiplication hardware," *Microprocessors and Microsystems*, vol. 78, pp. 103219, Oct. 2020.

[21] E. Ozturk, Y. Doroz, E. Savas, and B. Sunar, "A custom accelerator for homomorphic encryption applications," *IEEE Trans. Comput.*, vol. 66, no. 1, pp. 3–16, Jan. 2017.

[22] S. Kim, K. Lee, W. Cho, Y. Nam, J. H. Cheon, and R. A. Rutenbar, "Hardware architecture of a number theoretic transform for a bootstrappable RNS-based homomorphic encryption scheme," *2020 IEEE Annu. Int. Symp. Field-Programmable Custom Computing Machines (FCCM)*, Fayetteville, AR, USA, 03–06 May 2020, pp. 56–64.