Bootstrapping on SEAL

HyungChul Kang
Samsung Advanced Institute of Technology
Korea, Republic of
hc1803.kang@samsung.com

Joon-Woo Lee
Seoul National University
Korea, Republic of
joonwoo3511@ccl.snu.ac.kr

Yongwoo Lee
Seoul National University
Korea, Republic of
yongwool@ccl.snu.ac.kr

Young-Sik Kim
Chosun University
Korea, Republic of
iamyskim@chosun.ac.kr

Jong-Seon No
Seoul National University
Korea, Republic of
jsno@snu.ac.kr

ABSTRACT

We implement bootstrapping of RNS-CKKS on SEAL, a homomorphic encryption library released by Microsoft. And we measure the accuracy of encrypted data after bootstrapping for various parameters, which allows us to do more than thousands of homomorphic operations.

KEYWORDS

Bootstrapping, SEAL, Cheon-Kim-Kim-Song, RNS-CKKS

1 CKKS and RNS-CKKS

Cheon-Kim-Kim-Song (CKKS)[1] is a scheme proposed by Cheon et al.. The biggest feature of CKKS scheme is that it accepts errors as part of a message, unlike other homomorphic encryption schemes. Because of this feature, CKKS scheme can perform homomorphic operations among real numbers and complex numbers, which has attracted attention in the field of machine learning. RNS-CKKS scheme[2] is a residue number system (RNS) variant of CKKS scheme, where power-of-two moduli of rings are replaced with products of big primes by allowing the approximate rescaling. Currently, RNS-CKKS scheme is implemented in simple encrypted arithmetic library (SEAL)[3] released by Microsoft.

2 Bootstrapping

Since the modulus of a ciphertext in CKKS (or RNS-CKKS) scheme is continuously reduced as the several homomorphic multiplications with rescaling procedure are evaluated, the modulus eventually becomes too small to allow additional rescaling procedure at some time. Then, it becomes the smallest modulus q_0 from the initial modulus Q. Thus, we have to raise the modulus of the ciphertext at that time without modifying the message so that the additional rescaling procedure can be performed, which corresponds to the bootstrapping. Bootstrapping of CKKS scheme[4] raises the modulus q_0 reduced through rescaling to a larger modulus Q'. Raising modulus Q' means raising to a higher level in RNS-CKKS scheme.

Bootstrapping of CKKS(or RNS-CKKS) scheme is performed in the following process.

- ModRaise: ModRaise is a process of raising the modulus q_0 to the modulus Q. Due to this process, coefficients of plaintext polynomial are in the form of $m(Y) + q_0 \cdot I(X)$ ($Y = X^{N/n}$, N = polynomial degree, n = number of message slots).
- SubSum: Because of a sparse packing, the coefficients of plaintext polynomial are in the form of $m(Y) + q_0 \cdot I(X)$. If SubSum is performed after ModRaise, these are in the form of $m(Y) + q_0 \cdot I(Y)$. If you haven't done the sparse packing, you don't need SubSum.
- CoeffToSlot: A homomorphic evaluation is an operation on message slots. However, we have to perform a modular operation on the coefficients of plaintext polynomial. Therefore, we need to put the coefficients into the message slots. CoeffToSlot is the process of doing this.
- Modular Reduction: Modular Reduction is a process of approximating the modular reduction on the coefficients of plaintext polynomial contained in the message slots.
- SlotToCoeff: SlotToCoeff is an inverse operation of CoeffToSlot, which is a process of returning the coefficients of plaintext polynomial contained in the message slots back to their original positions.

3 Implementation Results

We implement bootstrapping of RNS-CKKS scheme on SEAL v3.5.6 for various parameters (Table 1. and 2.) using the algorithm in [5, 6]. The various parameters are defined as follows.

- $\log N$: a log value of a polynomial degree
- $\log Q$: a log value of an initial modulus
- λ: a security level
- $\log p$: a log value of a scaling factor
- $\log q_0$: a log value of the smallest modulus
- ϵ: a half-width of each approximation region for a scaled cosine fucntion
- h: a hamming weight of a private key
- log slots: a log value of message slots
- Precision: a precision of bootstrapping (integer(bit) + decimal(bit))
- The number of remained levels: remained levels after bootstrapping
- Time (sec): bootstrapping time per all message slots
- Amortized time (ms): bootstrapping time per one message slot

Table 1. Sets of parameters in SEAL

No.	$\log N$	$\log Q$	λ	$\log p$	$\log q_0$	ϵ	h
1	16	1560	128	40	60	2^{-3}	64
2	16	1550	128	50	60	2^{-6}	64
3	16	1535	128	55	60	2^{-5}	64
4	16	1740	128	40	60	2^{-3}	96
5	16	1760	128	50	60	2^{-6}	96
6	16	1760	128	55	60	2^{-5}	96
7	16	1740	128	40	60	2^{-3}	128
8	16	1760	128	50	60	2^{-6}	128
9	16	1760	128	55	60	2^{-5}	128
10	16	1760	128	40	60	2^{-3}	192
11	16	1770	128	50	60	2^{-6}	192
12	16	1765	128	55	60	2^{-5}	192

Table 1 lists the twelve set of parameters for the simulation of the bootstrapping of SEAL. Table 2. shows the performance measurement results for each parameter (CPU: Intel(R) Core(TM) i7-9700K CPU @ 3.60GHz). In the previous, only the number between 0 and 1 was treated as a message without considering an integer part. However, since bootstrapping for a number greater than 1 is required depending on applications, the simulation is performed to restore the integer part.

We implement CoeffToSlot and SlotToCoeff proposed by Chen et al.[5], and Modular Reduction proposed Lee et al.[6].

Table 2. Performance of the bootstrapping of SEAL

No.	log slots	Precision*	The number of remained levels	Time (sec)	Amortized time (ms)
1	10	$17 + 6.44$	7	93.8	91.6
	11	$17 + 5.96$	7	107.3	52.4

	12	17 + 5.46	7	128.8	31.5
	13	17 + 4.97	7	156.3	19.1
	14	17 + 4.46	7	193.6	11.8
	15	17 + 4.17	9	338.6	10.3
2	10	4 + 17.11	8	86.1	84.1
	11	4 + 16.62	8	99.7	48.7
	12	4 + 16.13	8	120.1	29.3
	13	4 + 15.62	8	146.7	17.9
	14	4 + 15.11	8	182.5	11.1
	15	4 + 14.86	9	301.5	9.2
3	10	1 + 22.00	6	78.7	76.9
	11	1 + 21.49	6	90.9	44.4
	12	1 + 20.96	6	110.1	26.9
	13	1 + 20.47	6	134.8	16.5
	14	1 + 19.98	6	167.9	10.2
	15	1 + 19.70	7	273.4	8.3
4	10	17 + 4.63	10	117.5	114.8
	11	17 + 4.12	10	134.9	65.9
	12	17 + 3.59	10	160.4	39.2
	13	17 + 3.11	10	193.3	23.6
	14	17 + 2.59	10	237.9	14.5
	15	17 + 2.33	12	413.6	12.6
5	10	4 + 16.08	11	109.5	106.9
	11	4 + 15.55	11	125.7	61.4
	12	4 + 15.05	11	151.6	37.0
	13	4 + 14.57	11	183.8	22.4
	14	4 + 14.08	11	226.8	13.8
	15	4 + 13.80	12	372.6	11.4
6	10	1 + 21.03	9	100.5	98.1
	11	1 + 20.59	9	115.8	56.5
	12	1 + 20.06	9	138.6	33.8
	13	1 + 19.60	9	168.7	20.6
	14	1 + 19.07	9	207.9	12.7
	15	1 + 18.80	10	340.0	10.4
7	10	17 + 4.30	10	118.1	115.4
	11	17 + 3.64	10	135.9	66.4
	12	17 + 3.18	10	161.6	39.5
	13	17 + 2.66	10	195.9	23.9
	14	17 + 2.16	10	237.8	14.5
	15	17 + 1.89	12	415.3	12.7
8	10	4 + 14.19	11	110.9	108.3
	11	4 + 13.71	11	127.6	62.3
	12	4 + 13.17	11	153.5	37.5
	13	4 + 12.65	11	185.4	22.6
	14	4 + 12.17	11	227.9	13.9
	15	4 + 11.89	12	375.4	11.5
9	10	1 + 19.16	9	101.8	99.4
	11	1 + 18.65	9	116.7	57.0
	12	1 + 18.14	9	140.0	34.2
	13	1 + 17.69	9	170.3	20.8
	14	1 + 17.17	9	209.2	12.8
	15	1 + 16.91	10	344.7	10.5

10	10	17 + 5.30	9	117.7	114.9
	11	17 + 4.79	9	135.1	65.9
	12	17 + 4.31	9	159.8	39.0
	13	17 + 3.78	9	191.9	23.4
	14	17 + 3.29	9	234.4	14.3
	15	17 + 3.02	10	383.6	11.7
11	10	4 + 15.30	10	110.8	108.2
	11	4 + 14.81	10	127.6	62.3
	12	4 + 14.29	10	152.2	37.2
	13	4 + 13.80	10	184.2	22.5
	14	4 + 13.27	10	226.6	13.8
	15	4 + 13.04	11	369.3	11.3
12	10	1 + 20.32	8	102.0	99.6
	11	1 + 19.80	8	117.0	57.1
	12	1 + 19.27	8	140.0	34.2
	13	1 + 18.82	8	169.5	20.7
	14	1 + 18.32	8	208.3	12.7
	15	1 + 18.05	9	340.6	10.4

* In the precision(accuracy) of the above table, a + b stands for the accuracy of the integer part, a and the accuracy of the fractional part, b.

ACKNOWLEDGMENTS

REFERENCES

[1] Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10624, pp. 409–437. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70694-8_15

[2] J. H. Cheon, K. Han, A. Kim, M. Kim, Y. Song, A full RNS variant of approximate homomorphic encryption, in SAC 2018, pp. 347–368, 2018

[3] Microsoft SEAL (release 3.5), https://github.com/Microsoft/SEAL, apr 2020, Microsoft Research, Redmod, WA.

[4] Cheon, J.H., Han, K., Kim, A., Kim, M., Song, Y.: Bootstrapping for approximate homomorphic encryption. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10820, pp. 360–384. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78381-9_14

[5] Chen, H., Chillotti, I., Song, Y.: Improved bootstrapping for approximate homomorphic encryption. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. LNCS, vol. 11477, pp. 34–54. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17656-3_2

[6] Lee, J.W., Lee, E., Lee, Y., Kim, Y.S., No, J.S., High-Precision Bootstrapping of RNS-CKKS Homomorphic Encryption Using Optimal Polynomial Approximation and Inverse Sine Function, in ePrint 2020/552.