Comparative Analysis of Homomorphic Encryption Algorithms Based on Learning with Errors

Mikhail Babenko
North-Caucasus Federal University,
Stavropol, Russia
mgbabenko@ncfu.ru

Elena Golimblevskaia
North-Caucasus Federal University,
Stavropol, Russia
elena.golimblevskaya@gmail.com

Egor Shiryaev
North-Caucasus Federal University,
Stavropol, Russia
ea_or@list.ru

***Abstract*— The widespread use of cloud technology allows to optimize the economic costs of maintaining the IT infrastructure of enterprises, but this increases the likelihood of theft of confidential data. One of the mechanisms to protect data from theft is cryptography. Using the classical primitives of symmetric and asymmetric encryption does not allow to process data in encrypted form. Homomorphic encryption is used for processing confidential data. Homomorphic encryption allows to perform arithmetic operations over encrypted text and obtain an encrypted result that corresponds to the result of operations performed over plain text. One of the perspective directions for constructing homomorphic ciphers is homomorphic ciphers based on Learning with Errors. In this article we examine the cryptographic properties of existing homomorphic ciphers (CKKS, BFV) based on Learning with Errors, compare their technical characteristics: cryptographic strength and data redundancy, data encoding and decoding speed, speed of arithmetic operations, data addition and multiplication, Key Switching operation speed.**

Keywords—Homomorphic encryption, FHE scheme, Residue Number System, Brakerski Fan Vercauteren scheme, Cheon Kim Kim Song scheme, cyclotomic ring, LattiGo, GoLang

I. INTRODUCTION

For large enterprises of any line of business, for smooth operation, it is necessary to maintain information channels, as well as timely processing and transmission of information. Currently, cloud technologies are often used to solve this problem. They allow storing information in one place giving access to several users simultaneously. To maintain the confidentiality of information (personal information of users, information that is a corporate secret, etc.) transmitted within the network, cloud services must provide a high level of cryptographic stability. However, the encryption and decryption of information takes a fairly high percentage of the computing power of the network, the increased load on which can have a negative impact on the operation of the entire network. The fact that homomorphic encryption allows performing calculations (changing) of information without having to decrypt it can increase performance and speed up the response time of the entire network, while maintaining a high degree of security.

Thus, the task is to find the most productive scheme of fully homomorphic encryption (FHE) [1] [2]. Based on the analysis of the HE [3], [4], [5], [6], [7], [8] schemes, we chose two for the research: Cheon, Kim, Kim and Song (CKKS) [12] and Brakerski/Fan-Vercauteren (BFV) [9], [10], [11]. Their main feature is the use of the residue number system (RNS) for performing operations in schemes. The study consists in measuring the execution time of the main functions in the schemes, thereby determining the most productive scheme. For the study, 4 sets of parameters are taken, which determine the dependence of performance on the degree of required security.

II. BACKGROUND

For our research, schemes were selected that perform completely homomorphic encryption, and also have an Residue Number System (RNS) implementation [14].

A. Homomorphic encryption

Fully homomorphic encryption is a form of encryption, which has to satisfy the additional requirement of homomorphism with respect to any operations on plaintexts. The encryption function $E(k,m)$, where m is a plaintext, k is an encryption key, is homomorphic with respect to the operation * on plaintexts, if and only if there is an effective algorithm M that receives any pair of ciphertexts of the form $E(k,m_1), E(k,m_2)$, which then produces a ciphertext c such that when decrypting c, plaintext $m_1 * m_2$ will be obtained [15]. A homomorphism with respect to the operation + is defined similarly. When partially homomorphic cryptosystems are homomorphic with respect to only one plaintext operation, fully homomorphic systems support homomorphism with respect to both operations (both addition and multiplication) [16]. This means that the following conditions are fulfilled for them:

$$\begin{cases} Dec\big(Enc(m_1) \otimes Enc(m_2)\big) = m_1 * m_2 \\ Dec\big(Enc(m_1) \oplus Enc(m_2)\big) = m_1 + m_2 \end{cases} \quad (1)$$

In this case, homomorphism in addition and multiplication operations is sufficient for the system to be completely homomorphic [16].

It is this feature of FHE that allows us to talk about the performance growth we are claiming, the ability to perform these operations on encrypted text can improve the performance of both cloud storage services and cloud computing centers.

B. Residue Number System

RNS is a weighted number system based on modular arithmetic [17]. The representation of a number in RNS is based on a comparison of two integers modulo and the Chinese Remainder Theorem (CRT). Thus, RNS can be defined as a set of coprime modules $(\rho_1, \rho_2, \dots, \rho_n)$, whose vector is called the base of RNS, as well as its product $P = \rho_1 \cdot \rho_2 \cdot \dots \cdot \rho_n$, and so that every integer X that belongs to the range $[0, P-1]$ is associated with a set of residuals $(\alpha_1, \alpha, \dots, \alpha_n)$, where

$$\begin{aligned} &\alpha_1 \equiv X (mod\ \rho_1) \\ &\alpha_2 \equiv X (mod\ \rho_2) \\ &\dots\dots\dots\dots\dots\dots \\ &\alpha_n \equiv X (mod\ \rho_n) \end{aligned} \quad (2)$$

Also, based on the corollary of CRT, the uniqueness of the representation of non-negative integers from the interval $[0, P-1]$ is guaranteed [18].

In this case, the main advantage of representing numbers in the RNS is determined by the fact that such operations as addition, subtraction and multiplication can be performed by the formula:

$$A * B = (\alpha_1, \alpha_2, \dots, \alpha_n) * (\beta_1, \beta_2, \dots, \beta_n) = \\ = ((\alpha_1 * \beta_1) \bmod \rho_1), ((\alpha_2 * \beta_2) \bmod \rho_2), \dots, ((\alpha_n * \beta_n) \bmod \rho_n), \quad (3)$$

where * denotes operations such as addition, multiplication or subtraction [19]. These operations are called modular [19], since for them to be performed in RNS, one processing cycle of numerical values is sufficient. Moreover, this processing takes place in parallel, and the value of the number in each category is independent of other categories.

III. Homomorphic Encryption Schemes

To conduct research on HE schemes, we selected the LattiGo lattice-based cryptography library [13], written in the Golang language. This library contains a set of functions that implement homomorphic encryption schemes. The LattiGo structure allows performing various studies on schemes, to experiment with both the complete schemes and individual operations performed in these schemes. All schemes meet publicly available safety standards. The parameters of it are presented below.

$N = 2^{\log N}$: the ring dimension. It determines cyclotomic polynomial degree. Moreover, it is the amount of polynomials coefficients in both plaintext and ciphertext. It can be only a power of two. N influence both security and performance. Because of this, for the schemes to work correctly, setting the parameter N requires special attention.

Q: ciphertext module. In LattiGo, it is selected as the product of small, relatively simple q_i modules that provide $q_i \equiv 1 \bmod 2N$, which makes representation in RNS possible. The choice of size of q_i modules is in the range from 50 to 60 bits, which gives better performance. Q affects security and performance at the same time; if N is fixed and Q is bigger, this means reduced security and performance. Q is connected with N and must be carefully selected.

σ: variance, that is used for error polynomials. It influences security of the scheme.

It is also worth noting that all the schemes presented in this paper have a common HE base. Schemes perform arithmetic over the plaintext and ciphertext spaces. The plaintext space and the ciphertext space share a cyclotomic ring, which we denote in this paper as $\mathcal{H}$.

$$\mathcal{H} = \mathbb{Z}_Q[X]/(X^N + 1) \quad (4)$$

where N is a power of two.

Based on the arithmetic used by the scheme, various batch coding is performed. However, the mapping of arrays of numbers in a polynomial occurs with the property inherent in all HE schemes presented in this paper:

$$decode(encode(m_1) \otimes encode(m_2)) \approx m_1 \odot m_2 \quad (5)$$

where $\otimes$ as a component-wise product, and $\odot$ as a nega-cyclic convolution.

A. CKKS Scheme

CKKS is a homomorphic encryption scheme for approximate number arithmetic, suggested by Cheon, Kim, Kim, and Song. This scheme can be used for arithmetic over complex numbers and has RNS implementation. Batch encoding of this scheme is comparable to a cyclotomic ring as:

$$\mathbb{C}^{N/2} \leftrightarrow \mathcal{H}, \quad (6)$$

Moreover CKKS depends on such parameter as plaintext scale. The values have to be scaled in order to be encoded on polynomial with integer coefficients. This parameter affects the accuracy of the output and the maximum permissible depth for the security parameter used. The configuration of parameters for CKKS is very dependent on the application, requiring a preliminary analysis of the scheme, performed in encrypted form, which in certain operations can reduce the scheme performance.

B. BFV Scheme

The Fan-Vercauteren version of scale-invariant Brakerski HE scheme is also considered in this paper. LattiGo contains RNS-implementation of this scheme. The scheme provides arithmetic over $\mathbb{Z}_t^N$. Then the batch coding of this scheme, similar to the CKKS scheme, will consist of the following:

$$\mathbb{Z}_t^N \leftrightarrow \mathcal{H}. \quad (7)$$

If CKKS had only one independent parameter, then the BFV scheme has the following set of independent parameters:

P : expanded ciphertext module, which is used exclusively for the Mul operation (multiplication) and the like, with no effect on the degree of security. It is also defined as the product of small, relatively simple modules p_j and must be selected in such a way that $P > Q$ with a small margin (~ 20 bits), which is realized by means of one smaller Q module.

t: plaintext module. This module determines the value that is maximally possible for the plaintext coefficient. If the calculation leads to a higher value, then the value decreases modulo plaintext. For batching to work, the value must be simple and satisfy $t \equiv 1 \bmod 2N$. This module also does not affect security.

IV. Analysis of HE Schemes Funrion Performance

Benchmarking is performed for functions such as Encoding, Decoding, Encryption, Decryption, Addition, Multiplication, Relinearization and KeySwitching. Measurements are carried out for different dimensions of the encrypted vector (from 128 to 2048 numbers) and for different encryption parameters (Table 1) [20].

TABLE 1 Security Parameters

ID	$\log_2 N$	$\log_2 Q$	σ
1	12	109	3.2
2	13	218	3.2
3	14	438	3.2
4	15	881	3.2

A. Function Performance

1) Performance of Encoding Function

Measuring the performance of the Encoding function shows the superiority of the BFV scheme. The results were obtained in the experiment, due to which it can be noted that the performance of both schemes in this function does not significantly depend on the dimension of the encrypted vector. But as security options increase, the CKKS scheme requires more time for the Encoding function. This is due to the fact that in the CKKS scheme the encrypted vector consists of complex numbers, and encoding it in plaintext with integer coefficients requires more time than the similar procedure in the BFV scheme, which uses an integer vector.

TABLE 2 ENCODING TEST RESULTS, *ns*

	ID	Dimension of encrypted data vector				
		128	**256**	**512**	**1024**	**2048**
CKKS	1	308963	333249	375870	460903	554734
	2	1873608	1973020	1884327	2002769	2215958
	3	6234737	6972733	6540090	7255575	7366968
	4	24415039	26627552	27666560	27404332	25905446
	ID	**Dimension of encrypted data vector**				
		128	**256**	**512**	**1024**	**2048**
BFV	1	211498	181524	169412	181010	171326
	2	496768	484148	511326	358980	472566
	3	1177199	1382577	959615	1301481	1067257
	4	3020121	3607336	3947700	2737189	3241723

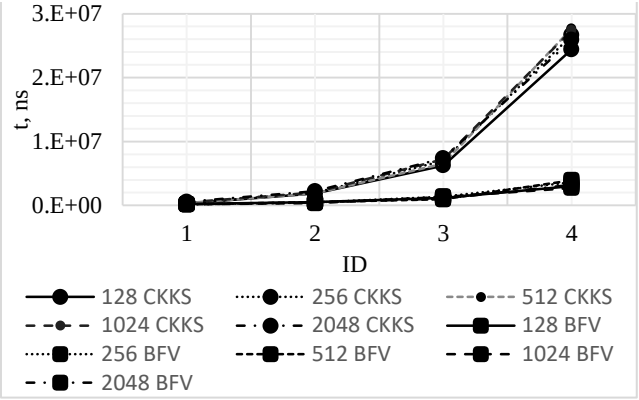

Fig. 1. Encoding time for BFV and CKKS

2) Performance of Decoding Function

The Decoding function depends on the dimension of the vector. The CKKS scheme scales worse due to the fact described above. The BFV scheme demonstrates the best speed of this function, but its advantage is negligible.

TABLE 3 DECODING TEST RESULTS, *ns*

	ID	Dimension of encrypted data vector				
		128	**256**	**512**	**1024**	**2048**
CKKS	1	1018856	1285869	1625036	2418919	4052705
	2	4200838	5038371	5152239	7399088	10315995
	3	13104508	14706263	14633906	17983167	21821681
	4	46158396	57014194	52938040	58693620	60117011
	ID	**Dimension of encrypted data vector**				
		128	**256**	**512**	**1024**	**2048**
BFV	1	1154013	1175882	1048750	1274209	1075534
	2	3151799	4236402	3288824	4564273	3038342
	3	12881134	16357558	14754335	13649037	12698237
	4	42709982	39141703	59220495	44049092	41901074

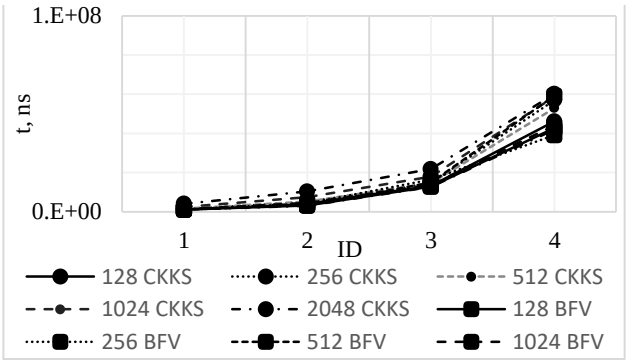

Fig. 2. Decoding time for BFV and CKKS

3) Performance of Ecryption Function

Both encryption options (with public key and private key) have approximately the same performance on the first three sets of parameters. With the highest security requirement, in the case of secret key encryption, the BFV scheme scales worse and has lower performance. In the case of public key encryption, the results are reversed. In general, BFV is inferior in the performance of this function, This situation is explained by the peculiarities of arithmetic implementation, which were described in section 3.B.

TABLE 4 ENCRYPTION WITH SECRET KEY TEST RESULTS, *ns*

	ID	Dimension of encrypted data vector				
		128	**256**	**512**	**1024**	**2048**
CKKS	1	2334489	2103892	2226278	2179083	2173156
	2	9766863	9382399	9218207	10142885	9507998
	3	33230264	33543617	31445147	32506263	33135191
	4	127892000	114750689	120734989	113363544	118239467
	ID	**Dimension of encrypted data vector**				
		128	**256**	**512**	**1024**	**2048**
BFV	1	2641997	3559443	2930780	3024797	3233360
	2	7644061	7330153	7071365	6598306	9201644
	3	34327793	34766696	37137252	36247603	36192216
	4	87575015	145189343	104613338	97278158	147942129

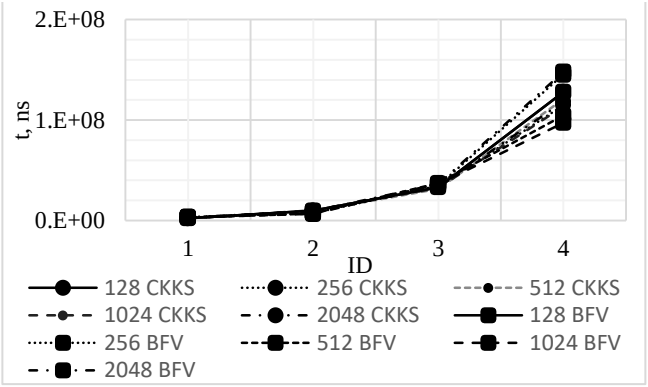

Fig. 3. Encryption Secret Key time for BFV and CKKS

TABLE 5 ENCRYPTION WITH PUBLIC KEY TEST RESULTS, *ns*

	ID	Dimension of encrypted data vector				
		128	**256**	**512**	**1024**	**2048**
CKKS	1	2958304	2999553	2931131	3075510	3193087
	2	12985601	13081973	13171425	12353057	13008715
	3	46876542	52181155	42546888	44072000	43259396
	4	188649729	162992743	185340700	153663429	164277171
	ID	**Dimension of encrypted data vector**				
		128	**256**	**512**	**1024**	**2048**
BFV	1	3504304	3091037	2876235	3835154	4615228
	2	9948796	12968238	11532753	12323740	12444451

	3	45251935	45050544	42973388	49087468	34707847
	4	124654010	127437450	177076771	168856400	172971357

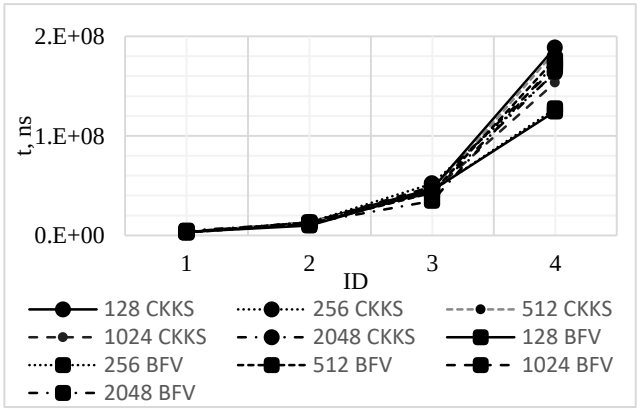

Fig. 4. Encryption Public Key time for BFV and CKKS

4) Performance of Decryption Function

When measuring the Decryption function, the results which are displayed in Table 6 were obtained. Due to the fact that the BFV scheme uses NTT at the decryption stage, which the CKKS scheme uses in the Decode function, this function takes much less time with the CKKS scheme.

TABLE 6 DECRYPTION TEST RESULTS, *ns*

	ID	Dimension of encrypted data vector				
		128	**256**	**512**	**1024**	**2048**
CKKS	1	75295	64490	91053	86014	83257
	2	704320	558029	500854	589965	569032
	3	1740359	1325015	1841163	1609382	1552320
	4	7188552	8656303	6451143	8596062	8852536
	ID	**Dimension of encrypted data vector**				
		128	**256**	**512**	**1024**	**2048**
BFV	1	1411973	1304798	1211739	1062796	893495
	2	4277947	3599959	3417547	3493461	3516575
	3	14734244	12412934	17919951	12627585	14222562
	4	79417746	81107353	59298720	81474821	82202793

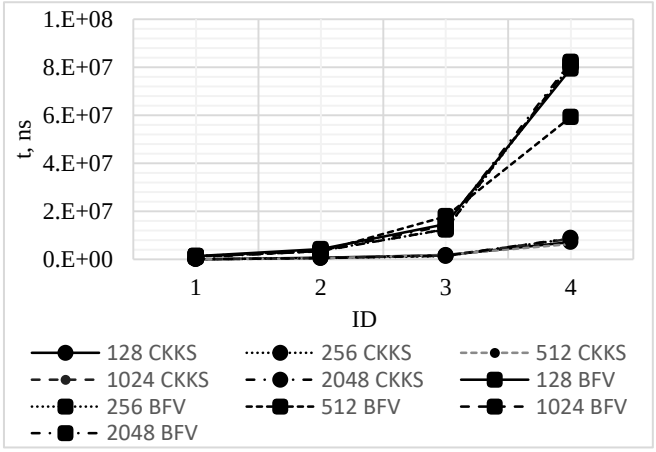

Fig. 5. Decryption time for BFV and CKKS

5) Performance of Addition Function

The study of the homomorphic summation function also shows the advantage of the BFV scheme. This is because in the CKKS scheme data is properly scaled before the operation.

TABLE 7 ADDITION TEST RESULTS, *ns*

	ID	Dimension of encrypted data vector				
		128	**256**	**512**	**1024**	**2048**
CKKS	1	24465	37828	39856	30617	47407
	2	145437	235402	291096	312227	299579
	3	589837	981959	1081942	1020950	1071134
	4	2692325	3453714	3695143	2700388	4009002
	ID	**Dimension of encrypted data vector**				
		128	**256**	**512**	**1024**	**2048**
BFV	1	35960	34953	36248	35226	31980
	2	159519	113622	158053	123147	161107
	3	630680	635675	658730	442162	667723
	4	1724961	2314835	1881167	2609253	2738442

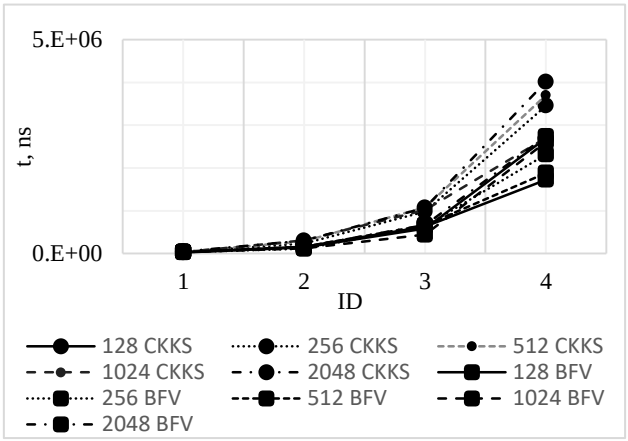

Fig. 6. Addition time for BFV and CKKS

6) Performance of Multiplication Function

Simulation of the multiplication of ciphertext by a scalar and by ciphertext showed two different results. When multiplied by a scalar, BFV scheme shows high performance. But the BFV scheme has scalar multiplication only for the uint64 type, while CKKS provides a solution for types like complex128, float64, uint64, int64 and int. The constant is represented as a complex number, when multiplying the ciphertext by which, obviously, it takes more time than when multiplying by a constant of the uint64 type in the BFV scheme.

TABLE 8 MULTIPLICATION BY SCALAR TEST RESULTS, , *ns*

	ID	Dimension of encrypted data vector				
		128	**256**	**512**	**1024**	**2048**
CKKS	1	42654	62583	70964	67642	63079
	2	231795	503098	545495	518164	550229
	3	1069628	1745948	1756096	1705708	1733524
	4	4077041	6304774	6041801	6270333	6130658
	ID	**Dimension of encrypted data vector**				
		128	**256**	**512**	**1024**	**2048**
BFV	1	48430	48057	43500	48024	45157
	2	219919	170531	217853	186792	213763
	3	854740	815247	649023	879803	839350
	4	2831192	3393249	2587298	2762533	2592716

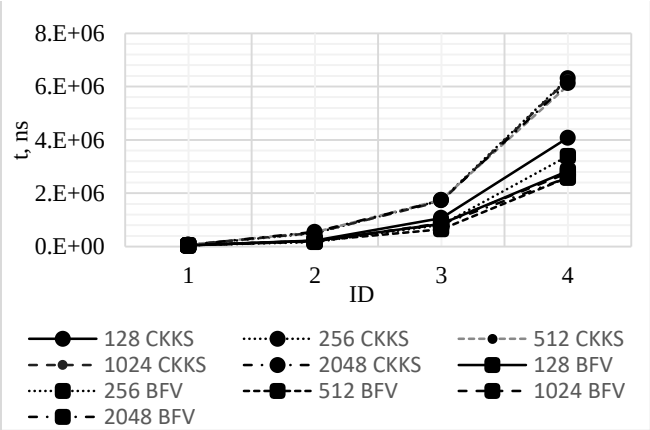

Fig. 7. Multiplication on Scalar time for BFV and CKKS

At the same time, the multiplication of ciphertext by ciphertext in the CKKS scheme has higher performance, and also, compared to BFV, a relatively small increase in time spent with increasing security settings.

TABLE 9 MULTIPLICATION TEST RESULTS, *ns*

	ID	Dimension of encrypted data vector				
		128	**256**	**512**	**1024**	**2048**
CKKS	1	134293	216760	180038	171766	205594
	2	674873	1469598	1525148	1078809	1152720
	3	3782338	4913585	5253617	5065445	3949636
	4	13078065	12682334	17907985	15632646	17966423
	ID	**Dimension of encrypted data vector**				
		128	**256**	**512**	**1024**	**2048**
BFV	1	7089414	7487352	7695449	10207294	7628154
	2	31767778	25332290	35383221	37529547	33403094
	3	170244000	170229078	158613588	171963733	180915567
	4	932056250	967022400	925598350	556013800	951451000

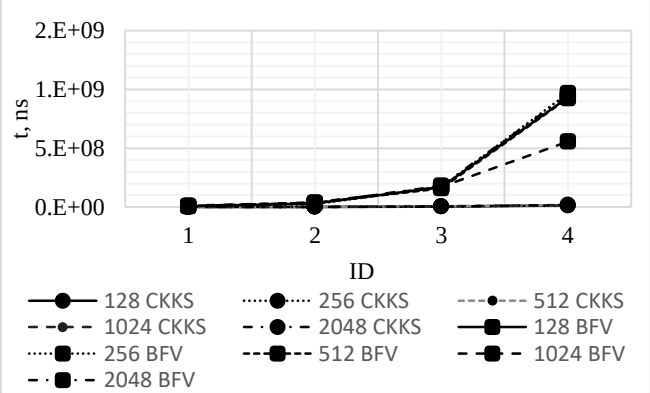

Fig. 8. Multiplication time for BFV and CKKS

7) Performance of the Relinearization Function

From Fig. 9 and Tab. 10 we can see that the Relinearization function in the BFV scheme with higher safety parameters is more productive. But on the first set of parameters, the implementation of the CKKS scheme wins.

TABLE 10 RELINEARIZATION TEST RESULTS, *ns*

	ID	Dimension of encrypted data vector				
		128	**256**	**512**	**1024**	**2048**
CKKS	1	1687438	2661698	3424449	2261363	3231668
	2	18201540	34499628	30833154	30375355	30167662
	3	100644400	154891650	153884971	103854875	146318000
	4	383689733	442374400	633888950	658028000	666219500
	ID	**Dimension of encrypted data vector**				
		128	**256**	**512**	**1024**	**2048**
BFV	1	3724441	2389712	3317893	4066252	3757963
	2	11464962	11950768	11067133	12411460	12308602
	3	69749413	70541320	75965527	61616068	75667007
	4	290493225	292239375	350729100	364713920	306935225

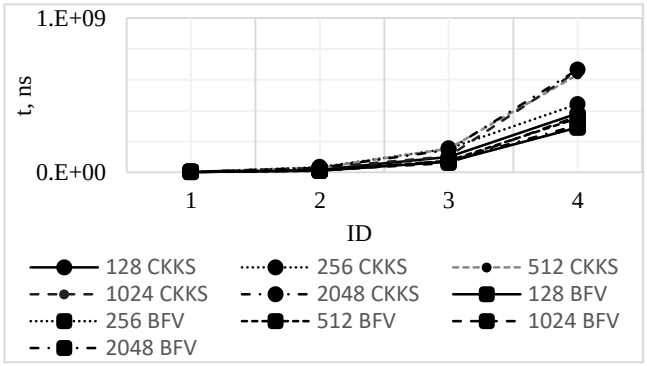

Fig. 9. Relinearization time for BFV and CKKS

8) Performance of KeySwitching Function

In this study, the CKKS scheme showed higher performance. This function is faster with complex numbers, not integers, since the complex number field is easier to scale.

TABLE 11 KEYSWITCHING TEST RESULTS, *ns*

	ID	Dimension of encrypted data vector				
		128	**256**	**512**	**1024**	**2048**
CKKS	1	2503032	3026201	3224305	3203122	2546295
	2	28739441	28862725	26122406	35456117	23952320
	3	135554285	115695133	100226020	138455820	133445758
	4	679806800	660336400	686261350	504191200	685178233
	ID	**Dimension of encrypted data vector**				
		128	**256**	**512**	**1024**	**2048**
BFV	1	2938459	3010961	3227951	2688819	3374234
	2	10792889	9802067	8450965	10495674	10323025
	3	60385157	65758147	55269293	53448786	41285438
	4	276312900	342915800	342230575	236232667	254161425

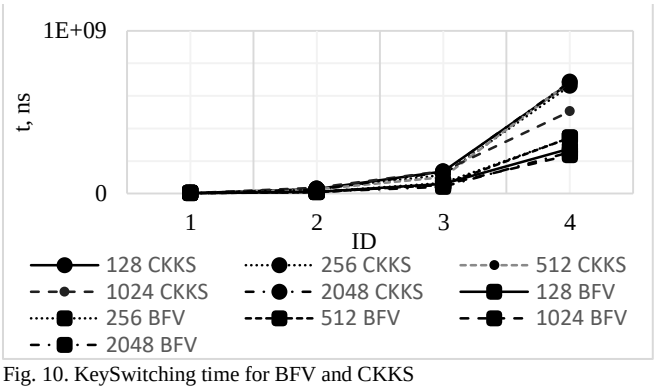

Fig. 10. KeySwitching time for BFV and CKKS

B. Signal ro Noise Ratio (SNR)

It is known that the ciphertext obtained with HE has redundancy. Thus, the degree of this redundancy is an important parameter of the HE schemes. This redundancy can be determined by examining SNR.

1) SNR of Plaintext to Vector

When conducting SNR of the plaintext to the initial vector, a high degree of redundancy was found in both schemes, but the CKKS scheme showed a better ratio.

TABLE 12 SNR OF PLAINTEXT TO VECTOR

	ID	Dimension of encrypted data vector				
		128	256	512	1024	2048
CKKS	1	1.049194	8.38208	4.185791	2.090729	1.04755
	2	2.093929	16.74509	8.37439	4.188866	2.092964
	3	5.538696	44.25195	22.10657	11.06628	5.530708
	4	11.06649	88.49725	44.25009	22.12093	11.06154
	ID	Dimension of encrypted data vector				
		128	256	512	1024	2048
BFV	1	39.62492	19.81713	9.907773	4.958288	2.474978
	2	164.451	82.67813	41.0744	20.5674	10.2885
	3	665.6461	333.0266	166.6605	83.25319	41.63363
	4	2838.382	1422.057	710.3599	354.8242	177.552

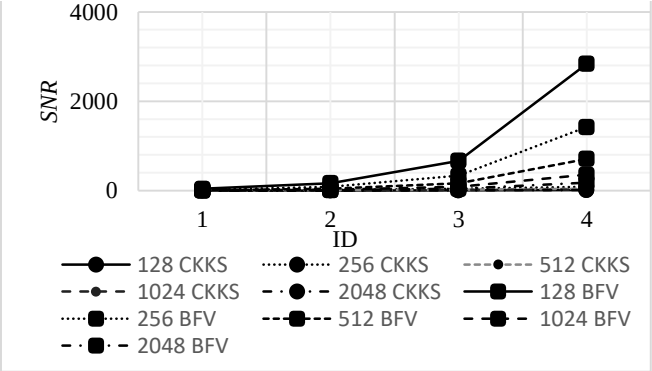

Fig. 11. SNR of Plaintext to Vector

2) *SNR of Ciphertext to Plaintext*

Within plaintext encryption, the noise in the CKKS scheme increases much more than in the BFV scheme. But with higher security settings, its increase is reduced.

TABLE 13 SNR OF CIPHERTEXT TO PLAINTEXT

	ID	Dimension of encrypted data vector				
		128	256	512	1024	2048
CKKS	1	20.30157	20.34031	20.34885	20.37499	20.34864
	2	20.35725	20.36439	20.36092	20.35067	20.36643
	3	16.05651	16.08642	16.1052	16.08278	16.08872
	4	16.08454	16.09035	16.09035	16.09291	16.09277
	ID	Dimension of encrypted data vector				
		128	256	512	1024	2048
BFV	1	1.9481	1.949288	1.949583	1.948348	1.948181
	2	1.963438	1.963153	1.962645	1.963236	1.962972
	3	1.963446	1.96309	1.963343	1.963505	1.963333
	4	1.965543	1.965856	1.965686	1.965566	1.965581

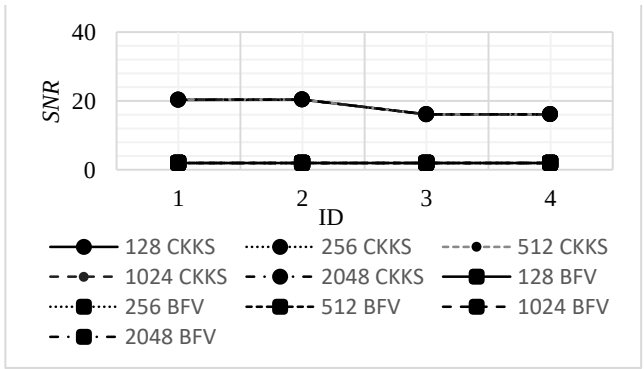

Fig. 12. SNR of ciphertext to plaintext

In general, it can be noted that with lower security settings and a larger dimension of the ciphertext, the BFV scheme shows less data redundancy. But in the case of high security settings, the CKKS scheme is clearly more profitable.

V. CONCLUSION

The use of cloud technology on the one hand can reduce the cost of maintaining IT infrastructure, but on the other hand has a number of limitations associated with the scope of application. For example, when processing confidential data, it is necessary to consider the risks of information theft, to reduce the probability of data theft, homomorphic encryption is used. In our work we investigate two schemes of CKKS and BFV homorphic data encryption from the point of view of technical characteristics, such as data encoding speed, data decoding speed, encryption and decryption speed, speed of arithmetic operations with encrypted texts, evaluates noise parameters that occur when encrypting data. A comparison of the two schemes shows that there is no one-size-fits-all approach that can be used as a universal solution. Further we plan to investigate a question of realization of matrix operations with use of various homomorphic encryption schemes.

ACKNOWLEDGMENT

The research was partially supported by the Russian Science Foundation Grant No. 19-71-10033.

REFERENCES

[1] Craig Gentry. A Fully Homomorphic Encryption Scheme. PhD thesis, Stanford University, 2009. crypto.stanford.edu/craig.

[2] Martin R. Albrecht, Shi Bai, and Léo Ducas. A subfield lattice attack on overstretched NTRU assumptions - cryptanalysis of some FHE and graded encoding schemes. In Matthew Robshaw and Jonathan Katz, editors, Advances in Cryptology – CRYPTO 2016, Part I, volume 9814 of Lecture Notes in Computer Science, pages 153–178, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany.

[3] Marten Van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully Homomorphic Encryption over the Integers. In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pages 24–43. Springer, 2010.

[4] Zvika Brakerski. Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP. In Advances in cryptology–crypto 2012, pages 868–886. Springer, 2012.

[5] Adriana L´opez-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-Fly Multiparty Computation on the Cloud via Multikey Fully Homomorphic Encryption. In Proceedings of the forty-fourth annual ACM symposium on Theory of computing, pages 1219–1234. ACM, 2012.

[6] Junfeng Fan and Frederik Vercauteren. Somewhat Practical Fully Homomorphic Encryption. IACR Cryptology ePrint Archive, 2012:144, 2012.

[7] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from Learning with Errors: Conceptually-Simpler, AsymptoticallyFaster, Attribute-Based. In Advances in Cryptology–CRYPTO 2013, pages 75–92. Springer, 2013.

[8] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) Fully Homomorphic Encryption without Bootstrapping. ACM Transactions on Computation Theory (TOCT), 6(3):13, 2014.

[9] TancredeLepointandMichaelNaehrig. AComparisonoftheHomomorphic Encryption Schemes FV and YASHE. In International Conference on Cryptology in Africa, pages 318–335. Springer, 2014.

[10] Jean-Claude Bajard, Julien Eynard, M Anwar Hasan, and Vincent Zucca. A Full RNS Variant of FV Like Somewhat Homomorphic Encryption Schemes. In International Conference on Selected Areas in Cryptography, pages 423–442. Springer, 2016.

[11] Shai Halevi, Yuriy Polyakov, and Victor Shoup. An Improved RNS Variant of the BFV Homomorphic Encryption Scheme. https://eprint.iacr.org/2018/117.

[12] Hao Chen Kim Laine and Rachel Player. Simple Encrypted Arithmetic Library-SEAL (v2.1). Technical report, Technical report, Microsoft Research, 2016.

[13] Lattigo: lattice-based cryptographic library in Go URL: https://github.com/ldsec/lattigo

[14] Behrooz Parhami, Computer Arithmetic: Algorithms and Hardware Design. 2nd edition, Oxford University Press, New York, 2010. 641+xxv p. ISBN 978-0-19-532848-6.

[15] Varnovsky N. P., Shokurov A. V. Homomorphic encryption // Transactions of ISP RAS. 2007.

[16] Babenko Lyudmila Klimentyevna, Burtyka Filipp Borisovich, Makarevich Oleg Borisovich, Trepacheva Alina Viktorovna Methods of fully homomorphic encryption based on matrix polynomials // Cybersecurity issues. 2015. No1 (9).

[17] Erdnieva N. S. The use of special modules of the system of residual classes for redundant representations // Vestnik ASTU. Series: Management, Computing and Informatics. 2013. No2.

[18] Sagalovich Yu. L. Introduction to algebraic codes - 2nd ed. - M.: IPPI RAS, 2010 .-- 320 p. - ISBN 978-5-901158-14-2

[19] Lavrinenko A.N., Chervyakov N.I. Research of non-modular operations in the system of residual classes // Scientific statements of Belgorod State University. Series: Economics. Computer science. 2012. No. 1-1 (120).

[20] Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, Satya Lokam, Daniele Micciancio, Dustin Moody, Travis Morrison, Amit Sahai, and Vinod Vaikuntanathan. Homomorphic encryption security standard. Technical report, HomomorphicEncryption.org, Toronto, Canada, November 2018.