Evaluating Performance Characteristics of Threshold Fully Homomorphic Encryption for Distributed Analytics Scenarios

Svetlana Boudko, Kristian Teig Grønvold
DART,
Norwegian Computing Center
Oslo, Norway
e-mail: svetlana@nr.no | kristegro@gmail.com

***Abstract*—Distributed analytics, such as federated learning, involve collaborative computation across multiple decentralized devices. This approach not only reduces data transfer costs but also offers some degree of protection for privacy-sensitive information. To achieve a higher level of privacy protection, it is recommended to use more advanced privacy-preserving technologies, such as homomorphic encryption. However, the use of homomorphic encryption schemes results in high computational costs. In this study, we evaluate the performance characteristics of threshold fully homomorphic encryption, a technique that can be effectively applied in multi-user environments and distributed analytics scenarios. We present results from the performance evaluation of the Cheon-Kim-Kim-Song scheme.**

Keywords-privacy; data security; threshold homomorphic encryption; multi-party computation; distributed analytics.

I. Introduction

Homomorphic encryption [1] is a form of encryption that allows computations to be carried out on ciphertext, generating an encrypted result which, when decrypted, matches the result of operations performed on the plaintext. This unique property of homomorphic encryption makes it highly valuable in the field of data privacy and distributed analytics, e.g. federated learning, where sensitive data are processed.

The first practical Fully Homomorphic Encryption (FHE) scheme was proposed by Craig Gentry in 2009 [2]. Subsequently, various homomorphic encryption schemes have been introduced in the literature, all aiming to enhance computational efficiency [3]–[6]. Initially, these schemes were proposed as single-key homomorphic encryption methods. While these schemes are useful for several scenarios, they are not suitable for distributed analytics. In distributed analytics, different clients need their own unique secret keys to ensure protection of their data, making single-key systems inapplicable.

This problem is attempted to be addressed through Multi-Key Homomorphic Encryption (MKHE) [7]–[9], where each client holds its own secret key. However, current MKHE schemes are not yet practical for most applications due to their high computational cost. The key generation is computationally expensive, and the size of the generated ciphertext increases in proportion to the number of clients.

Threshold homomorphic encryption [10]–[13] is another multi-key scheme that addresses the issue of ciphertext expansion. As a result, it can be effectively utilized in distributed analytics. Currently, several standardization bodies, such as the National Institute of Standards and Technology (NIST), the International Organization for Standardization (ISO), and Homomorphic Encryption Standardization, have initiated their efforts on threshold cryptography and homomorphic encryption with the goal of establishing guidelines and recommendations for threshold cryptosystems and promoting wider adoption of these technologies [14]–[16]. While extensive research has been done exploring this approach, its practical application remains limited thus far. Hence, it is important to assess the practical applicability of threshold homomorphic encryption schemes.

In our previous work [17], we outlined the main components and directions for implementing privacy-preserving federated learning using threshold homomorphic encryption. In this ongoing study, we have evaluated a number of parameters to understand the practicality of this approach. These parameters include the size of the keys used in the threshold process, the runtime differences between computations on encrypted and plaintext data, and the key generation runtime for varying multiplicative depths.

The remainder of the paper is organized as follows. After presenting an overview of related work in Section II, we discuss a representative scenario in Section III. Evaluation setup and results are given in Section IV, before discussing future work and concluding in Section V.

II. Related work

The threshold multi-key encryption methods are based upon Learning With Errors (LWE) problem [18] and its more efficient version the Ring-LWE problem [19]. These problems belong to the category of lattice-based cryptography, also known to be post-quantum resistant.

In these methods, each party contributes a portion of the encryption key, and a specific threshold number of parties must be established before the data can be decrypted.

The clients generate their own secret key shares and collaborate to generate evaluation and joint public keys. The evaluation keys are sent to the server to perform calculations on encrypted data, and the generated joint public key is shared among the participants and used for data encryption. Each data owner encrypts their data using the joint public key, and the result is computed by the server in encrypted form using the evaluation keys. The clients collectively decrypt the result using their own secret key shares.

Several threshold multi-key homomorphic encryption schemes have been introduced in the literature and are available as open-source libraries [20], [21]. The Brakerski-Gentry-Vaikuntanathan (BGV) [3] and the Brakerski/Fan-Vercauteren (BFV) [4], [22] schemes rely on the Ring-LWE problem. The BGV scheme optimizes homomorphic operations by effectively managing the ciphertext's noise, primarily through enhancing the modulus switching technique. The BFV scheme is a scale-invariant construction with the same noise growth as in the BGV scheme. Both schemes are designed to support computations over integer arithmetic circuits.

The Ducas-Micciancio (FHEW) [23] and the Chillotti-Gama-Georgieva-Izabachene (CGGI) [5] schemes support the encryption of small bit-width integers and are constructed for Boolean circuit evaluation. In the FHEW scheme, the authors introduced a new bootstrapping technique that reduces the noise level. The CGGI scheme achieves faster bootstrapping by implementing programmable bootstrapping, which is a computational operation on a ciphertext performed during bootstrapping. This reduces the noise while processing ciphertexts.

Another threshold homomorphic encryption scheme, known as the Cheon-Kim-Kim-Song (CKKS) scheme [24], features approximate homomorphic computations over real and complex numbers. This scheme uses a rescaling operation to reduce noise growth from multiplications. Due to its support for arithmetic operations on real or complex numbers, the CKKS scheme is particularly well-suited to tackle a wide range of data analytics problems and is therefore chosen for the purpose of this study.

III. Representative Scenario

To demonstrate the applicability of threshold homomorphic encryption schemes, we consider a federated learning scenario where a group of clients collaboratively participates in the training and updating of machine learning models. Federated learning is a distributed machine learning approach that enables on-device model training using client-specific data, with further aggregation of the obtained local model updates on a central server, as depicted in Figure 1. Instead of sending data for centralized processing, this data is used locally to train the model. Subsequently, the model updates are sent to the central server to refresh the central model. The updated model is then sent back to the clients for the next update step.

Federated aggregation is a key process in federated learning. Cross-silo aggregation and cross-device aggregation are two concepts used in federated learning architectures. Cross-silo refers to the process of integrating, sharing, or collaborating on data and information across different departments or organizations (silos). On the other hand, cross-device aggregation involves the collection and integration of data from multiple devices, such as those in the Internet of Things (IoT).

These two approaches have different requirements. In the case of IoT devices, computing power and storage capacities are crucial. However, these parameters do not pose a challenge for institutions involved in cross-silo aggregation.

Federated averaging and weighted federated averaging are the most commonly used aggregation algorithms due to their efficiency [25]. In these methods, a subset of clients is selected to perform updates using stochastic gradient descent over several iterations. The process alternates between multiple local stochastic gradient updates and the exchange of their averaged weights for updates of the global model. Since these updates could potentially expose sensitive information and are susceptible to privacy attacks [26], [27], we employ homomorphic encryption to secure the data.

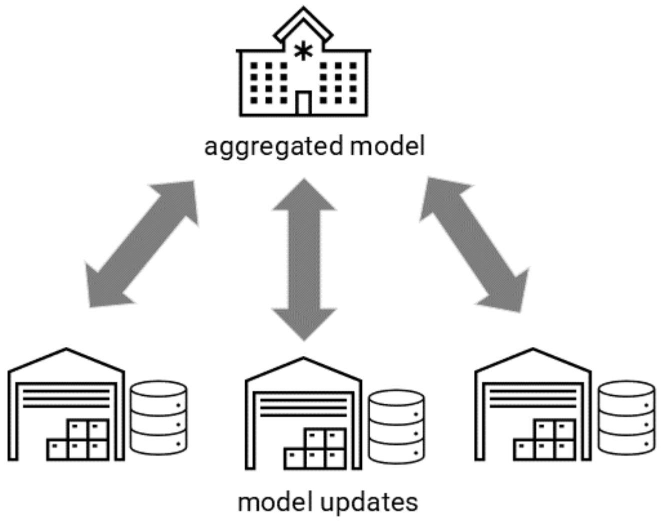

Figure 1. Cross-silo federated aggregation involving three clients and a central unit. The clients calculate their model updates using their own data and share these updates with the central unit. The central unit then generates a new model, which is sent back to the clients for the next iteration.

IV. Evaluation

A. Evaluation Setup

To perform the evaluation, we used the OpenFHE library [20]. This library is implemented in C++ and includes Python bindings, which simplify its integration with machine learning and data analytics platforms. The library supports threshold FHE for BGV, BFV, and CKKS schemes.

In our evaluation, we utilized the CKKS scheme. This scheme supports computations over real numbers and, therefore, can serve as a basis for developing data protection mechanisms for distributed analytics applications

B. Evaluation Results

Both key generation and general computations have been evaluated. Key generation was assessed for sets of 3, 4, 5, 6, 7, and 8 keys, across multiplicative depths of 5, 10, 15, and 20. The multiplicative depth refers to the maximum number of sequential multiplications that can be performed. Table I below shows the runtime for key generation in seconds at different multiplicative depths.

A larger multiplicative depth increases the time required to generate the same amount of keys. Additionally, it enlarges the size of the serializations of the cryptocontext and various keys, introducing further overhead for most applications since the

TABLE I. RUNTIME FOR DIFFERENT MULTIPLICATIVE DEPTHS.

Number of keys generated	Runtime in seconds MultDepth 5	MultDepth 10	MultDepth 15	MultDepth 20
3	12.28	17.02	48.93	61.67
4	14.98	22.80	64.58	82.07
5	18.23	27.88	81.28	101.52
6	21.52	33.59	98.63	123.64
7	25.11	39.02	112.34	141.71
8	28.55	44.47	129.62	163.07

cryptocontext and keys must be deserialized before use. Storage space might also become an issue, as each user must store the cryptocontext, the joint public key, the multiplication key, and their own secret key. See Table II below for a breakdown of file sizes at various multiplicative depths. Due to the costs

TABLE II. FILE SIZES FOR DIFFERENT MULTIPLICATIVE DEPTHS.

Multiplicative depth	5	10	15	20
Cryptocontext	21 KB	30 KB	40 KB	49 KB
Joint public key	41 MB	66 MB	189 MB	238 MB
Multiplication key	131 MB	209 MB	600 MB	757 MB
Secret key	14 MB	23 MB	65 MB	84 MB

associated with higher multiplicative depths, it is recommended to keep them as small as possible. Even if large computations are needed, it is possible to save on multiplicative depth by employing more efficient techniques, such as computing powers of two. OpenFHE also supports bootstrapping to reduce the depth of a ciphertext; however, the bootstrapping process itself requires some available depth and further increases runtime.

Evaluation has been done for computing averages, both on plaintext and encrypted data, using both weighted and unweighted approaches. Ten datasets were used, each containing from 1 to 10 elements, with values ranging from 1 to 100. Across all tests, the sum of the values was 2976, and the total number of elements across the datasets was 55. Table III presents a breakdown of the runtime in seconds and the estimated precision for the encrypted results.

TABLE III. RUNTIME AND PRECISION ESTIMATION FOR PLAINTEXT AND CIPHERTEXT

Operation	Runtime (s)	Estimated precision (bits)
Average on ciphertext, total	11.71	42
Average on ciphertext, computations only	1.19	
Average on plaintext	0.017	N/A
Weighted average on ciphertext, total	15.43	28
Weighted average on ciphertext, computations only	6.29	
Weighted average on plaintext	0.017	N/A

The runtime on encrypted data does not include the time needed to generate the keys, which are assumed to have been generated in advance. The runtime marked with 'computations only' also does not include the time needed to load the cryptocontext and keys, nor the time used for encryption and decryption. Thus, while the runtime is higher for encrypted data, much of the increase comes from other processes that are not directly related to the computation itself.

Both types of averages utilize the OpenFHE function EvalMult, and the weighted average also utilizes the function EvalDivide. EvalMult takes a ciphertext and either another ciphertext, a plaintext, or a constant, and computes the product of these. This computation is done per element for inputs that contain more than one element. Meanwhile, EvalDivide takes a ciphertext and computes its inverse.

In the unweighted average, EvalDivide is not used since the number of elements in a dataset is not encrypted. Based on the runtime results shown in Table III, it can be concluded that EvalDivide is slower than EvalMult. Moreover, using EvalDivide negatively affects the estimated precision of the results: the unweighted average has an estimated precision of around 42 bits, while the weighted average has an estimated precision of about 28 bits.

The precision of EvalDivide can be improved by increasing the degree parameter; however, this also increases the runtime. Additionally, a larger degree parameter requires a higher multiplicative depth, which, as previously noted, increases overhead for most applications.

V. CONCLUSION AND FUTURE WORK

This paper introduces an ongoing study that utilizes threshold fully homomorphic encryption to protect sensitive data within the context of distributed analytics applications. It presents preliminary results from the performance evaluation of the CKKS scheme, as implemented in the OpenFHE library. The aim was to evaluate the efficiency of computing averages and weighted averages for federated aggregation on encrypted data.

The results show that due to a large size of cryptocontext data and time required for encryption and decryption, applying this method is challenging for cross-device aggregation. For IoT devices, which have limited processing power and memory, handling large cryptocontext data can be unfeasible. Cross-silo scenarios, on the other hand, involve the collaboration of various institutions where processing power and storage capabilities do not pose a bottleneck. Therefore, they can effectively apply these methods.

Future work will involve several steps, including: (1) further design and analysis of extended scenarios and use cases; (2) development of a testing platform to evaluate the applicability of threshold homomorphic encryption schemes to various scenarios; and (3) analysis, implementation, and testing of communication protocols, mechanisms, and key generation processes.

Distributed analytics and homomorphic encryption require significant computational resources and may be slower compared to conventional methods. As the number of devices and the volume of data grow, scaling these technologies presents a substantial challenge. Research is needed to develop methods for scalable, decentralized learning and efficient homomorphic encryption. Therefore, a more comprehensive analysis and evaluation of available threshold fully homomorphic encryption schemes and libraries will be conducted, alongside integration with existing federated aggregation methods, and modification of these methods if required.

ACKNOWLEDGMENT

This work has been carried out in the context of the Center for Research-based Innovation NORCICS, funded by the Research Council of Norway, grant number 310105/F40.

REFERENCES

[1] R. L. Rivest, L. Adleman, and M. L. Dertouzos, “On data banks and privacy homomorphisms”, *Foundations of Secure Computation, Academia Press*, pp. 169–179, 1978.

[2] C. Gentry, “Fully homomorphic encryption using ideal lattices”, in *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, ser. STOC ’09, Bethesda, MD, USA: Association for Computing Machinery, 2009, pp. 169–178, ISBN: 9781605585062. DOI: 10.1145/1536414.1536440.

[3] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, “(leveled) fully homomorphic encryption without bootstrapping”, in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ser. ITCS ’12, Cambridge, Massachusetts: Association for Computing Machinery, 2012, pp. 309–325, ISBN: 9781450311151. DOI: 10.1145/2090236.2090262.

[4] Z. Brakerski, “Fully homomorphic encryption without modulus switching from classical gapsvp”, in *Advances in Cryptology – CRYPTO 2012*, R. Safavi-Naini and R. Canetti, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 868–886, ISBN: 978-3-642-32009-5.

[5] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, “Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds”, in *Advances in Cryptology – ASIACRYPT 2016*, J. H. Cheon and T. Takagi, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 3–33, ISBN: 978-3-662-53887-6.

[6] C. Gentry, A. Sahai, and B. Waters, “Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based”, in *Advances in Cryptology – CRYPTO 2013*, R. Canetti and J. A. Garay, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 75–92, ISBN: 978-3-642-40041-4.

[7] A. López-Alt, E. Tromer, and V. 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*, ser. STOC ’12, New York, New York, USA: Association for Computing Machinery, 2012, pp. 1219–1234, ISBN: 9781450312455. DOI: 10.1145/2213977.2214086.

[8] M. Clear and C. McGoldrick, “Multi-identity and multi-key leveled fhe from learning with errors”, in *Advances in Cryptology – CRYPTO 2015*, R. Gennaro and M. Robshaw, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 630–656, ISBN: 978-3-662-48000-7.

[9] Z. Brakerski and R. Perlman, “Lattice-based fully dynamic multi-key fhe with short ciphertexts”, in *Advances in Cryptology – CRYPTO 2016*, M. Robshaw and J. Katz, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 190–213, ISBN: 978-3-662-53018-4.

[10] D. Boneh *et al.*, “Threshold cryptosystems from threshold fully homomorphic encryption”, in *Advances in Cryptology – CRYPTO 2018*, H. Shacham and A. Boldyreva, Eds., Cham: Springer International Publishing, 2018, pp. 565–596, ISBN: 978-3-319-96884-1.

[11] Y. Desmedt, “Threshold cryptography”, in *Encyclopedia of Cryptography and Security*, H. C. A. van Tilborg and S. Jajodia, Eds. Boston, MA: Springer US, 2011, pp. 1288–1293, ISBN: 978-1-4419-5906-5. DOI: 10.1007/978-1-4419-5906-5_330.

[12] B. Schoenmakers, “Threshold homomorphic cryptosystems”, in *Encyclopedia of Cryptography and Security*, H. C. A. van Tilborg and S. Jajodia, Eds. Boston, MA: Springer US, 2011, pp. 1293–1294, ISBN: 978-1-4419-5906-5. DOI: 10.1007/978-1-4419-5906-5_13.

[13] R. Bendlin and I. Damgård, “Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems”, in *Theory of Cryptography*, D. Micciancio, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 201–218, ISBN: 978-3-642-11799-2.

[14] *Multi-party threshold cryptography*, Online: https://csrc.nist.gov/projects/threshold-cryptography, accessed: 2024-10-01, National Institute of Standards and Technology, 2024.

[15] *ISO/IEC 18033-6:2019, IT Security techniques – Encryption algorithms – Part 6: Homomorphic encryption*, Online: https://www.iso.org/standard/64772.html, accessed: 2024-10-01, International Organization for Standardization, 2019.

[16] M. Albrecht *et al.*, “Homomorphic encryption security standard”, HomomorphicEncryption.org, Toronto, Canada, Tech. Rep., Nov. 2018.

[17] S. Boudko, “Towards implementation of privacy-preserving federated learning aggregation using multi-key homomorphic encryption”, in *2024 Congress in Computer Science, Computer Engineering, & Applied Computing (CSCE)*, In press, 2024.

[18] O. Regev, “On lattices, learning with errors, random linear codes, and cryptography”, in *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, ser. STOC ’05, Baltimore, MD, USA: Association for Computing Machinery, 2005, pp. 84–93, ISBN: 1581139608. DOI: 10.1145/1060590.1060603.

[19] V. Lyubashevsky, C. Peikert, and O. Regev, “On ideal lattices and learning with errors over rings”, in *Advances in Cryptology – EUROCRYPT 2010*, H. Gilbert, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 1–23, ISBN: 978-3-642-13190-5.

[20] A. A. Badawi *et al.*, *OpenFHE: Open-source fully homomorphic encryption library*, Cryptology ePrint Archive, Paper 2022/915, accessed: 2024-10-01, 2022.

[21] *Lattigo v5*, Online: https://github.com/tuneinsight/lattigo, accessed: 2024-10-01, Nov. 2023.

[22] J. Fan and F. Vercauteren, *Somewhat practical fully homomorphic encryption*, Cryptology ePrint Archive, Paper 2012/144, accessed: 2024-10-01, 2012.

[23] L. Ducas and D. Micciancio, “Fhew: Bootstrapping homomorphic encryption in less than a second”, in *Advances in Cryptology – EUROCRYPT 2015*, E. Oswald and M. Fischlin, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 617–640, ISBN: 978-3-662-46800-5.

[24] J. H. Cheon, A. Kim, M. Kim, and Y. Song, “Homomorphic encryption for arithmetic of approximate numbers”, in *Advances in Cryptology – ASIACRYPT 2017*, T. Takagi and T. Peyrin, Eds., Cham: Springer International Publishing, 2017, pp. 409–437, ISBN: 978-3-319-70694-8.

[25] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, *Federated learning of deep networks using model averaging*, accessed: 2024-10-01, 2016. arXiv: 1602.05629.

[26] V. Mothukuri *et al.*, “A survey on security and privacy of federated learning”, *Future Generation Computer Systems*, vol. 115, pp. 619–640, 2021, accessed: 2024-10-01, ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2020.10.007.

[27] R. Gosselin, L. Vieu, F. Loukil, and A. Benoit, “Privacy and security in federated learning: A survey”, *Applied Sciences*, vol. 12, no. 19, 2022, ISSN: 2076-3417. DOI: 10.3390/app12199901.