FP32 Homomorphic Encryption and Central Server Commitment Schemes for Federated Learning

December 15, 2021

Abi Panday, Justin Mack, Mohib Jafri, Zeel Patel
{abishrantpanday, justinmack, mjafri, zeelpatel}@college.harvard.edu

***Abstract*—Privacy preservation is a critical consideration in the implementation of federated learning towards applications involving vulnerable data, like healthcare. This paper thus proposes two advancements towards this paradigm: floating-point homomorphic encryption and novel central server aggregation methods for the Paillier cryptosystem. The former employs both RSA and Paillier in order to encrypt the FP32 representation, eliminating the need for floating-point to integer conversions. This method is experimentally observed to have similar asymptotic runtime as the Python-Paillier implementation while having no encryption/decryption loss. The latter consists of two theoretical constructions that are tested against a baseline Paillier heuristic, with experimental results showing similar testing accuracy. These results, which can be extended to utilize different PHE schemes, provide a groundwork for making efficient PHE in federated learning a reality.**

I. Introduction

In recent years, the explosion of low-cost cloud computing services has enabled consumer adoption of complex machine learning protocols that leverage large volumes of data. A natural use case of this decentralized computing power is federated learning [1], which reduces the bottleneck of local hardware by allowing a large number of devices to train a shared model in a decentralized fashion.

One field that stands to benefit greatly from this paradigm is healthcare, where A.I. models have shown efficacy in classifying diseases based on images and biomarkers. In particular, with hospitals existing on a broad spectrum of local computation power, federated learning brings the promise of utilizing vast amounts of local data that otherwise couldn't be trained on. A major roadblock for widespread adoption, however, are regulations on the privacy and movement of patient data—as shown by [2], [3], effective attacks on federated learning exist that exploit the weight updates in each round.

This paper thus studies partially homomorphic encryption (PHE) as an avenue for efficient federated learning with privacy guarantees. Two large problems in PHE implementation within federated learning are tackled: the requirement for floating point numbers to be quantized into integers for encryption/decryption and the lack of central server aggregation that preserves homomorphic property. For the former, a novel mixed scheme for native homomorphic floating point encryption is constructed using RSA and Paillier. For the latter, two novel central server aggregation techniques are constructed. Quantization error is calculated and testing on CIFAR-10 classification is conducted, showing that the Paillier heuristics achieved better accuracy than baseline Paillier without requiring significantly more resources.

These results provide a way to perform homomorphic encryption, which works over integer rings, for the high precision floating point data required in machine learning. In addition, we illustrate the possibility of central server division methods even in additive homomorphic schemes, which allow for better training convergence without sacrificing on accuracy.

II. Problem to Solve

A. Problem Statement

This paper focuses on PHE as a defense against malicious backdoor attacks on federated learning that reconstruct private samples based on transmitted weights. Specifically, we look to solve two major problems towards effective PHE use within the field: integer quantization and central server averaging. The former arises because PHE schemes operate over finite fields such as $\mathbb{Z}_p$, meaning high-precision floating point numbers need to be quantized to integers, causing accuracy loss. The latter is a problem because PHE schemes by definition only operate over one kind of gate (either addition or multiplication). In our case, the central server is unable to send the average weight back to each agent under the Pailler cryptosystem to perform FedAVG. We thus present theoretical constructions to address these issues and implement them against baseline metrics to show their efficacy.

B. Project Goals

This paper has three novel objectives:

- **Construct native floating point homomorphic encryption:** implement a method to encrypt FP32 values with arbitrary precision such that the central server can perform homomorphic multiplication without quantization loss.
- **Construct privacy-preserving central server aggregation:** develop techniques to enable additively homomorphic central server aggregation that is close to an average while revealing no information about the participants in each round.
- **Measure accuracy loss:** evaluate the accuracy loss present in applying the baseline Paillier model as well as our theoretical heuristics. In addition, determine whether the heuristics have a significant impact on training time.

C. Evaluation Metrics

First, we measure the accuracy per round of the global federated learning model with a variety of encryption schemes and central server aggregation algorithms. Specifically, we measure accuracy for baseline encryption (no encryption) with FedAVG aggregation, Paillier encryption with Sum aggregation, Paillier encryption with Heuristic1 aggregation, and Paillier encryption with Heuristic2 aggregation. Finally, we encrypt and then decrypt a series of FP32 values with Paillier and with our novel mixed scheme in order to measure the difference in encryption time.

D. Theoretical Background

Definition 1 (Public Key PHE [4])**.** For a family of circuits $\mathcal{C}$, a $\mathcal{C}$-partially homomorphic encryption scheme is a tuple $(G, E, D, EVAL)$ of randomized algorithms with:

- **Key Generation:** $G(1^n)$ which takes in n and returns a public, private key pair (e, d).
- **Encryption:** $E_e(m)$ takes in key e, plaintext m and returns a ciphertext c.
- **Evaluation:** $EVAL_e(C, c_1, \dots, c_k)$ takes in key e, a circuit $C \in \mathcal{C}$, and a tuple $c_1, \dots, c_k$ of ciphertexts and previous evaluations and returns an output.
- **Decryption:** $D_d(c)$ takes in key d and ciphertext or evaluation result c and returns a plaintext.

We now introduce the Paillier encryption scheme, the treatment of which is based on the original paper [5]. For large primes p, q define $n = pq$ and $\lambda = \text{lcm}(p-1, q-1)$. In addition, consider the following notation:

Notation. An integer z is said to be an nth residue mod n^2 if there exists $y \in \mathbb{Z}_{n^2}^*$ such that $z = y^n \mod n^2$. Construct the set $S_n = \{0 \le u < n^2 : u = 1 \mod n$ and define the function L with domain S_n such that $L(u) = \frac{u-1}{n}$. Finally, let $\mathcal{B} \subset \mathbb{Z}_{n^2}^*$ be the set of elements whose order is divisible by n.

Definition 2 (Paillier Encryption Scheme)**.** Randomly select a base $g \in \mathcal{B}$; this can be done efficiently by checking if

$$\gcd(L(g^\lambda \mod n^2), n) = 1$$

The public key is (n, g), and the private key is λ. [1]

- *Encryption.* Given plaintext $m \in [0, n)$, select a random $r \in [0, n)$ The encryption of m is $c = g^m \cdot r^n \mod n^2$.
- *Decryption.* Given cyphertext $c \in [0, n^2)$. The decryption of c is
$$m = \frac{L(c^\lambda \mod n^2)}{L(g^\lambda \mod n^2)} \mod n.$$

Theorem. *The Paillier system is additively homomorphic.*

Proof. Let E, D be the encryption and decryption functions. For all plaintexts m_1, m_2 with base g, modulus n, and random r, we have

$$D(E(m_1) \cdot E(m_2)) = D\left(g^{m_1} r^n \cdot g^{m_2} r^n \mod n^2\right)$$
$$= D\left(g^{m_1 m_2} (r_1 r_2)^n \mod n^2\right)$$
$$= D(E(m_1 + m_2)) = m_1 + m_2 \mod n$$

III. Proposed Approach

First, we note that because encryptions and decryptions in our Paillier system occur over a 3072-bit modulus, normal methods of floating point quantization involve multiplying each FP32 value by b^i where the base $b = 16$ and the exponent i is adjusted for the required level of precision. The resulting integers do not map exactly back to the same FP32 values, causing model accuracy loss. Our novel approach utilizes RSA [6] and Paillier [5] encryption, which are multiplicatively and additively homomorphic respectively, as follows: Consider a 32-bit floating point number f represented with a 1-bit sign s, 8-bit exponent e, and 24-bit (one implicit bit) mantissa m. Let the overall encryption function be ENC and let E_r, E_p be RSA and Paillier encryption functions, respectively. To encrypt f, we follow the following procedure:

$$ENC(f) = \{E_r(s), E_p(e), E_r(m)\}$$

Recall that the multiplication of FP32 values is performed through addition of their exponents and multiplication of their signs and mantissas. Now, consider the product two FP32 values f_1, f_2 encrypted in this manner.

$$ENC(f_1) \cdot ENC(f_2)$$
$$= \{E_r(s_1) \cdot E_r(s_2)\}, \{E_p(e_1) \cdot E_p(e_2)\}, \{E_r(m_1) \cdot E_r(m_2)\}$$
$$= \{E_r(s_1 \cdot s_2)\}, \{E_p(e_1 + e_2)\}, \{E_r(m_1 \cdot m_2)\} = ENC(f_1 \cdot f_2)$$

Therefore, this encryption scheme is multiplicatively homomorphic for floating point values. This is superior to the existing quantization paradigm as it directly manipulates the FP32 representation and thus incurs no quantization loss. From a performance perspective, this approach removes the costly large number exponentiation and multiplication used traditionally to encode FP32 values as integers. However, it also requires multiple encryptions per value. With the correct bit interpretation, this approach could be easily modified to combine the sign and mantissa encryptions, but it still requires at least two encryptions.

In addition, on the Paillier encryption for federated learning front, our novel contribution consists of two theoretical constructions for approximate-division in the additively homomorphic paradigm, which are detailed in Section V. These will enable the central server to send back results that are closer to the average—as opposed to the current standard of using the whole encrypted sum to update the model—which we expect will improve convergence, although at the potential cost of additional time per epoch.

IV. Intellectual Points

This project provides two methods of improving federated learning with PHE: first by constructing a method by which FP32 numbers can be encrypted and multiplied without incurring quantization loss and then by enabling averaging in an additively homomorphic scheme. Both of these contributions

[1] One can check that (p, q) can be easily recovered from n, λ.

are theory-based and can be extended into different use-cases and different PHE schemes with slight modifications. In addition, the Paillier averaging constructions are compelling from a theoretical standpoint because they illustrate a general method of building approximate divisions (and potentially n^{th} roots) into schemes that otherwise would not allow them. Finally, the testing results show that these constructions even without further optimization perform in the expected manner, reducing encryption time and improving accuracy at the cost of computation power. Thus, this work presents a rich avenue for further explorations into the efficacy of PHE in federated learning and similar privacy-focused tasks.

V. Work Performed

A. Mixed Encryption

We developed a Python library called MixCrypt to implement the floating point encryption scheme described above. This library provides an encrypt function that takes as input a PyTorch tensor of type torch.float32 and returns the encrypted sign, exponent, and mantissa as a tuple. To accomplish this, we first view the FP32 values as uint32 values instead, and then use bit masking and bit shifting operations to extract integers representing the sign, exponent, and mantissa bitstrings. From there, we pass these integers to the RSA, Paillier, and RSA encryption algorithms, respectively, and return the encrypted results. MixCrypt also provides a decrypt function, which works similarly but in reverse. It takes as input the encrypted signs, exponent, and mantissa, and returns a PyTorch tensor of type torch.float32. We chose to work with tensors as our input and output to make integration with federated learning models easy. With this implementation, one can pass in the weight tensors stored in the model's state dictionary instead of being forced to loop over the individual values.

B. Central Server Aggregation

Since additions are performed mod n, under the right conditions we can perform divisions. More precisely, for Heuristic 1, suppose that $\gcd(k,n) = 1$ and $x \in [0, nk)$ is divisible by k. This ensures that $x/k = (k^{-1} \bmod n) \cdot x \mod n$. Similarly if $-nk < x < 0$, then we have

$$x/k = \big((k^{-1} \bmod n) \cdot x \mod n\big) - n.$$

It is important that we assume x is divisible by k, for otherwise we cannot guarantee that $x/k \in \mathbb{Q}$ is close to $x \cdot (k^{-1} \bmod n) \mod n$.

To use this strategy, for a round in which k agents participate, the central server sends the agents some multiple αk. Then, before sending their update, each agent first rounds their update to the nearest multiple of αk. This ensures that the sum of the updates is divisible by k, and so the mean of the updates can be correctly computed by the central server.

The above strategy, while hiding the true number of agents participating in each round, gives additional information to the agents. We thus propose the following construction, Heuristic 2, to perform a weighted average of the updates, which weighs each update within a multiplicative factor of 2.

Let m be maximum number of agents that can participate in a round and let d be the smallest positive integer satisfying $2^d \geq r$. Before sending an update, each agent rounds their value to the nearest multiple of 2^d. Note that this means the number of agents must be $O(\log(n)/b)$, where b is the number of bits of accuracy desired.

Suppose the central server wants to aggregate k updates. Let d' be the smallest positive integer such that $2^{d'} \geq k$. Then $d' \leq d$ since $k \leq m$. Of the k updates, the central server randomly selects $2k - 2^{d'}$ of them to divide by $2^{d'}$, and the remaining $2^{d'} - k$ to divide by $2^{d'-1}$. These are then summed together. This is indeed a weighted average since

$$\frac{2k - 2^{d'}}{2^{d'}} + \frac{2^{d'} - k}{2^{d'-1}} = 1.$$

Although we have used base 2 for simplicity, this easily generalizes to other bases at the cost of loss of accuracy due to rounding errors being larger, but allowing for weights which are closer to one another.

C. Implementation

To support our analysis, we implemented a flexible federated machine learning model with support for easily configurable accumulation and encryption functions. We start with a ConvNet class that defines our convolutional model—9 convolutional layers of progressively increasing size, each using a BatchNorm2D sub-layer and the ReLU activation function. We also have an AdaptiveAlgPool layer and a linear classifier layer.

We then define a Device class which represents a local device in the federated learning network. Each device has an id, a dataset, a ConvNet instance, a local epochs parameter, an encryption scheme, and a few other hyperparameters. Devices sample randomly from their dataset to construct their training set. Each device has standard train and test methods which use their local training dataset to improve their local model, and then a global test dataset to evaluate their local model. Each device also has transmit and load methods. The transmit method encrypts the device's model weights using the encryption scheme and returns those encrypted values. This method is called by the central server to get the encrypted weight updates. The load method allows the central server to provide a device with a set of encrypted weights, which the device then decrypts and loads as their state dictionary.

The Server class represents the central server, and it has a copy of the global model weights, as well as a configurable aggregation function to use when receiving weight updates.

Finally, the driver code initializes the server and device instances, then begins the federated learning rounds. In each round, a number of devices are selected to participate. These devices train on their local training dataset, and when finished the server calls all of their transmit methods to get a set of encrypted weights. The server then merges these weights according to its accumulation function, and calls the load method for every device to set this new, encrypted set of weights as the global model. At the end of each round, we check the global model's test accuracy and record that data.

VI. Results and Discussion

As illustrated in Figure 1, the MixCrypt was slower in encryption/decryption but looked to scale linearly with tensor size, same as the standard Python-Paillier library. MixCrypt, however, doesn't have any loss in its encryption/decryption and the actual implementation can be sped up through further parallelizing and tensor-level encryption, meaning it has high potential for real-world use.

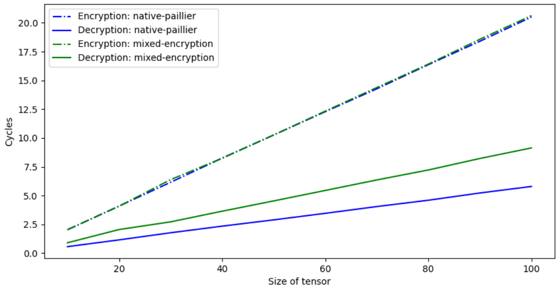

Fig. 1. Encryption and decryption time in CPU cycles for Python-Paillier and MixCrypt

Figure 2 then shows the test accuracy per epoch for the baseline FL model, along with baseline Paillier (with central server returning the sum), and the two heuristics. Of note is that the per-epoch training time compared to baseline Paillier was 23.7% longer for Heuristic 1 and 29% longer for Heuristic 2. From these results, we note that Heuristic 2 is an effective approximator for the average, with it being more accurate in the task than baseline Paillier. In addition, neither heuristic suffers a large computation penalty relative to the cost of using any encryption at all, meaning these methods, when optimized, will provide a better alternative to baseline Paillier.

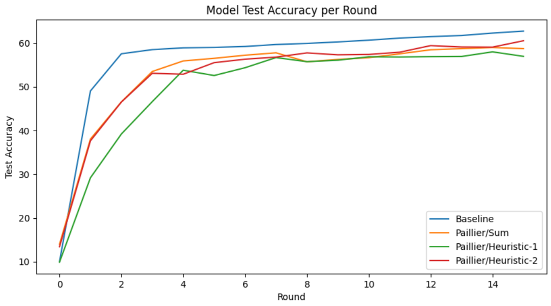

Fig. 2. Model test accuracy per round for different encryption schemes and accumulation functions

VII. Related Work

To motivate the need for encryption in federated learning, Enthoven and Al-Ars [2] propose a novel attack method named the *First Dense Layer Attack* (FIDEL), which is effective in reconstructing private training samples from weight updates provided to the central server for densely connected networks and convolutional neural networks. They show that the FIDEL attack can recover about 66% of private data samples from a client's model update with very little computational resources. In a similar vein, Wang et al. [3] explore malicious server attacks on specific agents in order to recover private data. They show user-level privacy leakage in a framework that incorporates generative adversarial networks (GANs) with a multi-task discriminator, which enables the generator to reconstruct user specified privacy data without interfering in the training process of federated learning.

Within the realm of partially homomorphic encryption, Fang et al. [7] and Muhammad et al. [8] provide accuracy and speed results for machine learning models. Muhammad et al. do this by implementing an on-device machine learning model that learns on input encrypted by the Paillier scheme and achieves 92.2% accuracy on the MNIST hand-written digit identification dataset. Fang et al., on the other hand, apply Paillier encryption to a multi-party federated learning protocol, showing the dependence of computational overhead on key length and accuracy deviation on classification tasks of 1%.

Jiang et al. [9], on the other hand, tackle the significant computation and communication overhead of homomorphic encryption schemes by proposing FLASHE, an encryption framework tailored for federated learning. FLASHE drops asymmetric key design and instead only performs modular addition with random numbers, vastly lowering the computational cost of each encryption. They find that FLASHE only has a 6% impact on training compared to unencrypted plaintexts, which is a substantial improvement over traditional PHE methods. Finally, Moon et al. [10] construct an FP32 encryption scheme for FHE systems for privacy-preserving data analysis. On the theoretical side, Zhang et al. [11] provide a method for transforming the high computational cost of homomorphic operations into a high communication complexity and Li et al. [12] explore another avenue for improving encryption/decryption efficiency: altering the underlying homomorphic framework.

VIII. Conclusion

Our work focused on advancing the practicality of PHE in federated learning by developing novel methods for central server aggregation during Paillier encryption and eliminating the need for floating-point conversions to integers in PHE. We showed that MixCrypt achieves a comparable asymptotic runtime to the standard Python-Paillier approach with no loss during encryption/decryption, opening the door for realistic implementations of PHE in federated learning to preserve privacy. Our theoretical constructions are also validated experimentally, where the second, more complex heuristic achieved better test accuracy, likely due to better weight convergence. Our constructions are novel within the literature, with only an analog of MixCrypt having appeared for FHE schemes [10]. We achieved the three broad goals initially scoped out and through our implementation process, we merged the gap between efficient theoretical results and practicable code. Future work in this area would involve parallelization of encryption for ML workflows, which was our major bottleneck, theoretical expansion of our heuristics to different partially homomorphic encryption schemes, and improvement of the MixCrypt algorithm to enable floating point addition.

IX. Contributions

All members of the team contributed equally to this research with: Abi taking leadership on identifying the research opportunity, developing the theoretical mathematical foundations for PHE, programming the heuristics, and wrote most of the paper. Justin helped develop the broader MixCrypt library, and the flexible federated learning model. Zeel contributed to developing the baseline model, multithreading the codebase so PHE could compute faster, and wrote various sections of the final report. Mohib executed many of the performance comparisons, constructed the graphs, programmed sections of the paillier optimization set, and wrote sections of the report.

References

[1] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, 2016.

[2] D. Enthoven and Z. Al-Ars, "Fidel: Reconstructing private training samples from weight updates in federated learning," *arXiv preprint arXiv:2101.00159*, 2021.

[3] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pp. 2512–2520, IEEE, 2019.

[4] F. Armknecht, C. Boyd, C. Carr, K. Gjøsteen, A. Jäschke, C. A. Reuter, and M. Strand, "A guide to fully homomorphic encryption.," *IACR Cryptol. ePrint Arch.*, vol. 2015, p. 1192, 2015.

[5] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *International conference on the theory and applications of cryptographic techniques*, pp. 223–238, Springer, 1999.

[6] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[7] H. Fang and Q. Qian, "Privacy preserving machine learning with homomorphic encryption and federated learning," *Future Internet*, vol. 13, no. 4, p. 94, 2021.

[8] K. Muhammad, K. A. Sugeng, and H. Murfi, "Machine learning with partially homomorphic encrypted data," in *Journal of Physics: Conference Series*, vol. 1108, p. 012112, IOP Publishing, 2018.

[9] Z. Jiang, W. Wang, and Y. Liu, "Flashe: Additively symmetric homomorphic encryption for cross-silo federated learning," *arXiv preprint arXiv:2109.00675*, 2021.

[10] S. Moon and Y. Lee, "An efficient encrypted floating-point representation using heaan and tfhe," *Security and Communication Networks*, vol. 2020, 2020.

[11] Q. Zhang, L. T. Yang, and Z. Chen, "Privacy preserving deep computation model on cloud for big data feature learning," *IEEE Transactions on Computers*, vol. 65, no. 5, pp. 1351–1362, 2015.

[12] J. Li, X. Kuang, S. Lin, X. Ma, and Y. Tang, "Privacy preservation for machine learning training and classification based on homomorphic encryption schemes," *Information Sciences*, vol. 526, pp. 166–179, 2020.