Enhancing Privacy in Machine Learning Services with Hybrid Homomorphic Encryption: Guard ML Framework

Mohammed Shamar Yadkar, Sefer Kurnaz

Altınbaş Üniversitesi, Istanbul, Turkey
Email: mohammedshamar17@gmail.com

Machine Learning (ML) stands as a pivotal field within data science, driving innovations across multiple sectors. However, The increasing risk of harmful assaults on ML models poses serious privacy concerns that could hinder its widespread use. By using Privacy-Preserving Machine- Learning (PPML) techniques, these dangers can be lessened, such as Homomorphic Encryption (HE), have been developed to protect sensitive data. Despite its potential, traditional HE faces inefficiencies, particularly in highly scalable applications. This paper presents a novel approach, Hybrid Homomorphic Encryption (HHE), which merges symmetric cryptography with HE to address these inefficiencies. We introduce the Guard ML framework, designed for end devices, in order to facilitate encrypted data classification while protecting the privacy of both the input data and the ML models. utilizing a case study of heart disease classification utilizing sensitive ECG data, our methodology shows how HHE can be practically applied. Our approach is feasible because, despite a small drop in accuracy compared to unencrypted inference, analysts, as well as end devices incur very little communication and computing expenses. Our work lays the groundwork for a future of machine learning that is more safe and more privacy-conscious, especially on end devices with limited resources, by effectively incorporating HHE into PPML.
Keywords: ML, PPML, HE, HHE, Secure Classification.

1. Introduction

Among the many branches of data science, machine learning (ML) has grown in prominence in recent years., driving innovations across numerous sectors through advancements in automation, decision-making, and predictive analytics. But the current advancement of the use of ML models has brought about privacy-related issues especially when dealing with personal data. There is information leakage since competitors and negative actors can capitalize on the existing weaknesses in ML systems to steal valuable or personal information.

Considering these problems, PPML has become a key research field. PPML guarantees privacy on the side of the user and secure use of the data in order to withhold any leakage of information. It is worthy of note that within the various PPML techniques, one called Homomorphic Encryption (HE) has the capacity to compute on data that has been encrypted without having to decrypt it first. HE makes computations like addition and multiplication possible directly on encrypted data without ever reading the actual data. To make operations on encrypted data for an unlimited number of times while providing authentic outputs, Gentry [1] designed one of the first FHE algorithms. Meanwhile then, several HE schemes, such as CKKS [2], TFHE [3], and BFV [4, 5], have been developed to enhance efficiency and applicability, especially in ML applications like Machine Learning as a Service (MLaaS) [6-13].

Computational efficiency as well as practicality are two of HE's biggest obstacles, especially in highly scalable situations, despite its promise. The widespread use of HE is hindered by the computational cost and the substantial ciphertext extension that frequently leads to huge ciphertexts. Researchers have resorted to HHE in an effort to alleviate these problems [14, 15]. By integrating symmetric cryptography using HE, HHE decreases communication overhead and ciphertext size, making HE more user-friendly.

The first process of an HHE scheme is the encryption using a symmetric key algorithm to encrypt the data. Then, HE technique is utilised to homomorphically encrypt the symmetric key. Both of the ciphertexts are received by the server and using the symmetric key the server encrypts both ciphertexts in order to make one homomorphic with the other. By doing so, they yield substantially smaller ciphertexts than what is seen in prior art HE schemes while solving two major problems of pure HE, namely, high multiplicative depth and expensive computational costs. Additional symmetric ciphers specifically designed to be HE-friendly have been derived and implemented by the researchers, including HERA/Rubato [16], [17], [18], [15], and more to increase the efficiency of HHE. The privacy issue in the Context of machine learning prediction is resolved with the help of HHE so as to remove barriers in implementing secure PPML models across the devices while exploring a new technology for a new concept.

2. Background

The area of Privacy-Preserving Machine Learning (PPML) has actively evolved over the last years with many suggestions to reduce threats affecting data privacy in Machine Learning. Some of these methods are; differential privacy, secure multi-party computation and specifically Homomorphic Encryption (HE).

HE has received a lot of attention due to the fact that it has the capability of performing computations on encrypted data without necessarily having to decrypt it. Gentry’s FHE scheme created the basis for He modern schemes commonly used in PPML applications, including TFHE, BFV, and CKKS. BFV, which is an enhancement over Gentry’s original scheme, eradicates expensive bootstrapping operations and can be categorized under Somewhat Homomorphic Encryption (SHE) that enables several operations on integer ciphertexts [5]. The CKKS scheme allows performing computations on floating-point data with a restricted number of operations; it was developed by Cheon et al. The TFHE scheme was proposed by Chillotti et al. , which improved bootstrapping operations’ effectiveness and allows for an unlimited number of operations between binary values [14]. These HE schemes have been incorporated in the ML applications with requiring different methods for the implementation of nonlinear activation functions. For example, in TFHE, LUT searches for non-linear activations are utilized [24], whereas in the case of BFV and CKKS, polynomial approximations are used [19, 23]. These schemes have yielded high-accuracy results in PPML with reputed implementations such as TAPAS [20] that utilize TFHE, FHE-DiNN [21], and works [22-224]. However, not all the settings are suitable for the implementation of HE schemes because of their high computational overhead and large ciphertext expansion, despite the possibilities they hold. For this reason, HHE has been investigated to receive answers regard to its application and challenges.

In the early specifications of HHE technologies, the member accounts depended on other symmetric ciphers such as AES. Yet because of its high multiplicative depth, AES was found wanting as far as HHE schemes were concerned [15]. Therefore, the research on symmetric ciphers was complemented as optimized for HHE, with reference to certain criteria such as the ciphertext expansion [24] and employing filter permutators [25]. In any case, it can be concluded that HHE has the potential for real-world PPML applications, although its practical applicability is rather limited and there are few implementations reported in the literature at the time of this writing. Some of the widely known HHE schemes are the HERA, Elisabeth, and the PASTA. CKKS, together with HERA, supports carrying out operations with floating-point objects [16]. Elisabeth is optimized for the use of the TFHE scheme [11], whereas PASTA is designed for the BFV for integer type of data [15]. To also come with specifications similar to those of Rubato, HERA’s authors also extended WMA. Table 1 offers a summary of the most important research works with emphasis on the contribution and use of several HE and HHE systems in PPML.

Table 1: presents an overview of important studies conducted (HE and HHE) on PPML.

Study	Scheme	Key Features	Application	Reference
Gentry (2009)	FHE	First fully homomorphic encryption scheme	Foundational work	[1]
Cheon et al. (201)	CKKS	Limited operations on floating-point data	ML applications	[2]
Chillotti et al. (2016)	TFHE	Efficient bootstrapping, bitwise operations	ML applications	[3]
BFV Scheme	SHE	Limited operations on integer ciphertexts	ML applications	[5]
HERA (2020)	HHE (CKKS)	Stream cipher for floating-point data	PPML applications	[17]
Elisabeth (2020)	HHE (TFHE)	Optimized for TFHE scheme	PPML applications	[18]
HCNN	BFV	High-accuracy results with polynomial approximations	PPML applications	[2]
POSEIDON	CKKS	Efficient polynomial approximations	PPML applications	[22]

2.1 Foundational Concepts in HE, HHE, and ML

Homomorphic Encryption (HE) is a cryptographic scheme represented by a quadruple of probabilistic polynomial-time (PPT) algorithms: Keygen, Enc, Dec, and Eval. The Keygen algorithm generates a public key, an evaluation key, and a private key using a security parameter (λ). The Enc algorithm encrypts a message (x) using the public key, resulting in a ciphertext (c). The Eval algorithm takes the evaluation key, a function (f), and a set of ciphertexts to produce a new ciphertext (cf). Finally, the Dec algorithm decrypts the ciphertext using the private key to retrieve the plaintext (x).

Hybrid Homomorphic Encryption combines HE with symmetric-key encryption, represented Keygen, Enc, Decomp, Eval, and Dec are the five PPT algorithms that work together. Keygen algorithm generates HE and SKE keys. The Enc algorithm involves generating a symmetric key, encrypting this key with HE (resulting in cK), and encrypting the message (x) to resulting in c . The Decomp algorithm uses the evaluation key, the symmetrically encrypted ciphertext, and the homomorphically encrypted symmetric key to produce a homomorphic encryption of the message (x). The Eval algorithm evaluates a function (f) on homomorphic ciphertexts, and the Dec algorithm decrypts the evaluated ciphertext using the private key to output the function's result on the plaintext message.

Machine Learning (ML) consists of algorithms that use training data to train a model (f(θ)). Training involves finding optimal parameters (θ) such as weights (w) and biases (b) to make accurate predictions. Once trained, the model can infer or predict outcomes for new, unseen data. In this work, HHE is leveraged to develop PPML protocol that ensure data and model privacy during the inference phase.

3. Methodology

3.1 Model of System

a) User: A set of all the users that will encrypt the data and another key generated is exclusive for the user only.

b) Many CSPs remain the parties that are responsible for collecting such data that is symmetrically encrypted from various users.

c) Analyst: This individual must be in possession of a machine-learning model and in a position to interpret the outcomes of progression of machine-learning operations on pre-encrypted data stored at CSP.

d) Decrypted information is obtained from the HE evaluation of the collected encrypted data in order to understand users’ data. The System Design for PPML is depicted in Figure 1.

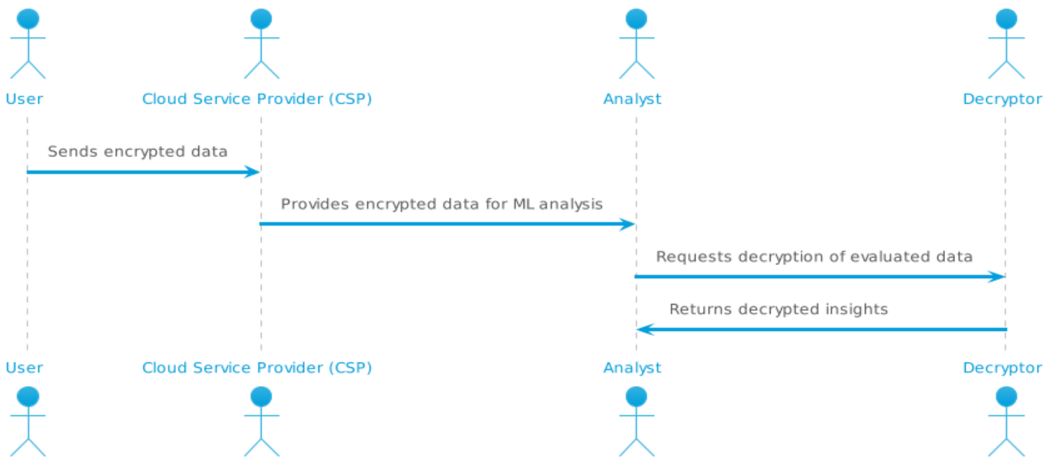

User
Cloud Service Provider (CSP)
Analyst
Decryptor
Sends encrypted data
Provides encrypted data for ML analysis
Requests decryption of evaluated data
Returns decrypted insights
User
Cloud Service Provider (CSP)
Analyst
Decryptor

Figure 1: System Model for Privacy-Preserving Machine Learning.

3.2 2GML Protocol in GuardML

This section explains the development of the 2GML protocol in the GuardML structure. The second phase, 2GML protocol, is a part of GuardML solution, being a part of the Hybrid Homomorphic Privacy-Preserving protocol, and it is addressed to specific machine learning applications needed for deploying in the commercial environment in which CSP owns the models. It keeps confidentiality on encrypted data and models and, at the same time, permits CSPs for computation. This setup is good when the analysts do not need to keep the model content but need CSP computational resources for some of the ML tasks.

The elements of the architectural design of the 2GML Protocol are Secure Symmetric Cipher (SKE), ABFV-based Hybrid Homomorphic Encryption (HHE), Public-Key Encryption Scheme (PKE), Signature Scheme (σ), Cryptographic Hash Function ($H(\cdot)$). These components facilitate security in encryption, decryption, signing, verification, and the message's integrity. Since they allow CSPs to own the ML models and offer backend computation, they are best suited to face commercial environments as depicted in Figure 2.

Moreover, Table 2 also gives a precise description of the main phases and operations of 2GML protocol described in GuardML, with focus on its core roles for supporting secure machine learning operations between Cloud Service Provider and a user.

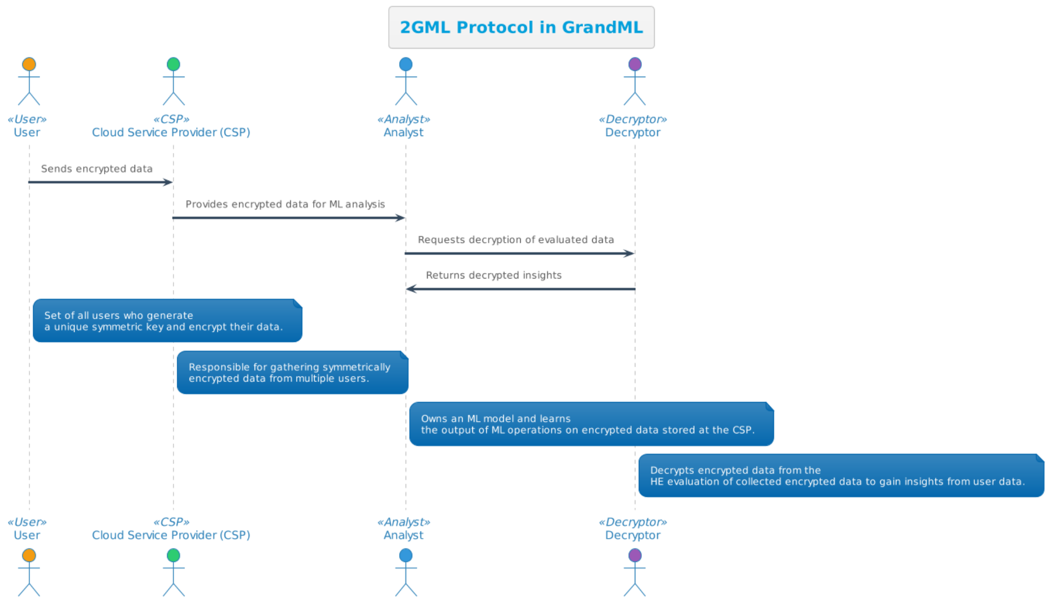

2GML Protocol in GrandML
«User» User
«CSP» Cloud Service Provider (CSP)
«Analyst» Analyst
«Decryptor» Decryptor
Sends encrypted data
Provides encrypted data for ML analysis
Requests decryption of evaluated data
Returns decrypted insights
Set of all users who generate a unique symmetric key and encrypt their data.
Responsible for gathering symmetrically encrypted data from multiple users.
Owns an ML model and learns the output of ML operations on encrypted data stored at the CSP.
Decrypts encrypted data from the HE evaluation of collected encrypted data to gain insights from user data.

Figure 2: 2GML Protocol: Building Blocks, Security Assumptions, and Use Case Suitability.

Table 2: Procedure Steps for the 2GML Method.

Phase	Description
2GML.Setup	• User uiu_iui generates HHE keys (pkui, skui, evkui) and shares pkui with CSP while sending evkui separately. • CSP generates its PKE key pair (pkCSP, skCSP). • User uiu_iui signs and CSP verifies the setup message m1m1m1.
2GML.Upload	• User uiu_iui encrypts data xix_ixi with SKE.Enc using a symmetric key Ki, producing ciphertexts cxicxicxi and cKicKicKi. • User uiu_iui homomorphically encrypts Ki into cKicKicKi with HHE.Enc. • User uiu_iui signs and CSP verifies the upload message m2m2m2.
2GML.Eval	• CSP decrypts cxicxicxi into c′xic'xic′xi with HHE.Decomp. • CSP uses c′xic'xic′xi, ML model parameters (w,bw, bw,b), and evkui in HHE.Eval to compute crescrescres. • CSP signs and sends crescrescres to uiu_iui in m3m3m3.
2GML.Classify	• User uiu_iui decrypts crescrescres with HHE. • Dec to obtain resresres, the prediction.

3.3 Attack Model and Analysis of SecurityThe capabilities of an opponent are the basis for the attack model that is utilized in the process of evaluating the security of GuardML, ADV, who can execute various attacks aimed at compromising protocol security and privacy. The cryptographic scheme used in GuardML has been rigorously tested and proven resilient against differential and linear statistical attacks, as well as algebraic attacks like Linearization and Gröbner Basis Attacks. The threat model focuses on communication between entities within the protocol, rather than the cryptographic scheme itself. ADV can corrupt any number of users and the Cloud Service Provider (CSP), mitigating the risk of basic man-in-the-middle attacks. Two potential attacks are identified: Ciphertext Substitution Attack and ML Model Unauthorized Access Attack. These attacks involve the malicious adversary replacing generated ciphertexts undetectable and gaining insights into the ML model used by the CSP or analyst without authorization.

3.4 Analysis of Security

This section focuses on the evaluation of the security of the GuardML protocol against potential attacks carried out by an adversary, referred to as ADV.

Table 3 presents a succinct summary of the security analysis performed on the 2GML protocol within GuardML. The document delineates the categories of attacks that are taken into account, together with their corresponding explanations, and the security guarantees offered by GuardML to counter these risks.

Table 3: Security Analysis of 2GML Protocol in GuardML

Attack Type	Description	Security Assurance
Ciphertext Substitution Attack	ADV attempts to replace genuine ciphertexts in 2GML.Upload or 2GML.Eval phases with indistinguishable fabricated ones.	EUF-CMA secure signature scheme (σ) ensures forgery resistance; negligible probability of success.
ML Model Unauthorized Access Attack	ADV colludes with users or compromises CSP to gain unauthorized access to the multi-layered ML model (f) used in 2GML.	Security relies on the complexity of multi-layered ML models and the semantically secure HE scheme.

4. Results

We evaluated encrypted inference in plaintext ECG data in floating-point and integer arithmetic across experiments with varying numbers of data inputs to assess the effectiveness the ecgPPML framework. Table 4 provides a summary of the findings, illustrating how the accuracy varied among different cases.

The encrypted inference methodology was first applied to a number of test split samples as part of our studies. We calculated the encryption test accuracy for each experiment by comparing the predictions made using the encrypted inferences to the ground-truth outputs. At the same time, we compared the outcomes of inferences performed on plaintext ECG data using floating-point and integer arithmetic. Table 5 shown analysis of accuracy, to delineates the accuracies achieved across different types of data representations (Plaintext Float, Plaintext Integer, and Encrypted) and varying numbers of data inputs:

a. Data Input: This column lists the number of data points used in each experiment, ranging from 1 to 2000, to assess the framework's performance under different data volumes.

b. Plaintext (Float): Represents accuracy percentages when data is in plaintext format using floating-point numbers. Accuracy begins at 100% for 1 data input and remains high, ranging between 86% and 88.2% as the number of inputs increases.

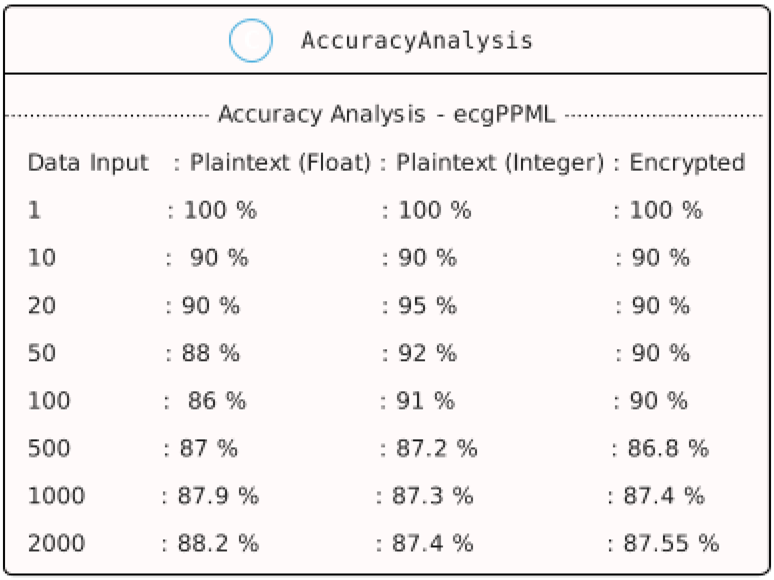
c. Plaintext (Integer): Shows accuracy percentages for plaintext data represented as integers. Starting at 100% accuracy for 1 input, the accuracy remains consistently high, peaking at 95% for 20 inputs and declining slightly to 87.2% to 95% for larger datasets (1000-2000 inputs).

d. Encrypted: Indicates accuracy percentages for data processed under encryption. Starting at 100% accuracy for 1 input, encrypted inference maintains robust performance, hovering around 90% accuracy even with larger datasets (86.8% for 500 inputs).

e. Plaintext Integer vs. Encrypted: The accuracies in integer arithmetic closely match those of encrypted inference, particularly evident when the input examples are limited (1-500). Both encrypted inference and plaintext integer exhibit small drops (0.5–0.8% lower) when the total amount of inputs increases (1000–2000), with encrypted inference occasionally surpassing plaintext integer accuracy by a minimal margin (0.1-0.15%).

The results prove the efficiency and reliability of the ecgPPML framework mainly focused on the high level of accuracy regardless the amount and type of data. The evaluation of encrypted inference is rather positive, as the accuracy loss in comparison with the plaintext techniques is minimal even with newly implemented homomorphic encryption noise that helps to make more correct predictions. In general, Table 4 delivers a wide panoramic view of how the ecgPPML addresses specific issues in privacy-preserving machine learning, produce robust performances and simultaneously guarantee the data protection, and therefore is fit for various real-world applications.

Table 4: Accuracy Analysis – ecgPPML



AccuracyAnalysis

Accuracy Analysis - ecgPPML

Data Input	Plaintext (Float)	Plaintext (Integer)	Encrypted
1	100 %	100 %	100 %
10	90 %	90 %	90 %
20	90 %	95 %	90 %
50	88 %	92 %	90 %
100	86 %	91 %	90 %
500	87 %	87.2 %	86.8 %
1000	87.9 %	87.3 %	87.4 %
2000	88.2 %	87.4 %	87.55 %

The effectiveness of 2GML for carrying out secure machine learning tasks is shown in the following figure, Figure 3. The combined outcome is the time taken to encrypt a symmetric key using the SKE technique applied in point (a). This duration rises with the input dimensions because more computations are required for each data point. When the inputs are many, the decryption during the assessment phase where EKCT is broken into sub-plaintexts also escalates significantly. Elliptic curve point multiplication takes the same amount of time for one input and multiple inputs based on the complexity of the computational problem, implying that there is efficient decryption of results from homomorphic encrypted outputs. The following figure shows the capability of protocol to perform computations on the encrypted data as well as tackle issues of performance which is a key factor in the development of secure data handling in sensitive applications.

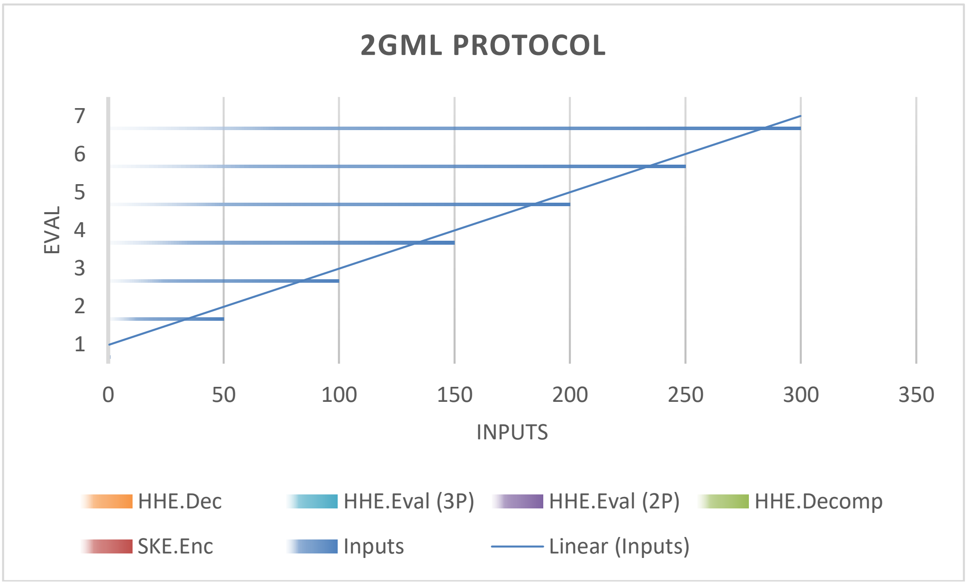
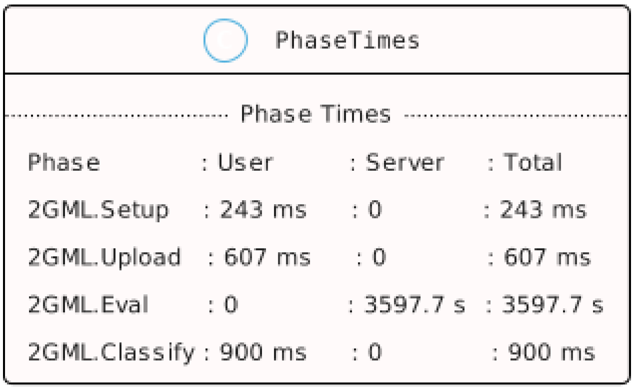


Figure 3: result of 2GML protocol.

Even though it's easy to use, the 2GML framework is very computationally intensive. Setup, data uploading, evaluation, and data classification are some of the steps involved. While the upload takes 607 milliseconds, the setup takes 243 milliseconds. The server needs 3597.7 seconds to complete the evaluation step, which includes processing 300 data inputs. On the user's end, the classification process takes 900 ms. Table 5 shows how these processes contribute to the 2GML framework's overall efficiency and speed.

Table 5: 2GML for 300 Data Inputs.



PhaseTimes

Phase Times

Phase	: User	: Server	: Total
2GML.Setup	: 243 ms	: 0	: 243 ms
2GML.Upload	: 607 ms	: 0	: 607 ms
2GML.Eval	: 0	: 3597.7 s	: 3597.7 s
2GML.Classify	: 900 ms	: 0	: 900 ms

5. Conclusion

In this paper, present the PPML technique is proposed through the utilization of HHE, which has been developed independently in this work. The solution is intended to provide optimum machine learning capability along with privacy to solve a wide range of PPML techniques and domains as pervasive computing. Implementing HHE, the authors were able to overcome the main challenges regarding application of real-life situations, namely data acquisition and management at the devices with limited computational resource such as IoT sensors and mobile appliances. The approach makes sure that solid security assurances are provided which almost do not affect the conduct of machine learning. The study also explains the synergy between cryptography and machine learning to support safe and efficient PPML services in these scenarios and other architectures regardless of being cloud-based, edge, or even constrained by resources. Thus, this approach provides new opportunities for building secure, private applications in such areas as healthcare, finance, smart cities, and others while maintaining data confidentiality and data integrity. The findings presented in the paper can be regarded as a groundwork for the subsequent developments in PPML to build efficient, high-quality, and secure machine learning systems.

References

1. Craig Gentry. 2009. A fully homomorphic encryption scheme. Stanford university.
2. Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. 2017. Homomorphic encryption for arithmetic of approximate numbers. In International Conference on the Theory and Application of Cryptology and Information Security. Springer, 409–437.
3. Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachene. 2016. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In Advances in Cryptology–ASIACRYPT 2016: 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I 22. Springer, 3–33.
4. Zvika Brakerski. 2012. Fully homomorphic encryption without modulus switching from classical GapSVP. In Annual Cryptology Conference. Springer, 868–886.
5. Junfeng Fan and Frederik Vercauteren. 2012. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive (2012).
6. Tanveer Khan, Alexandros Bakas, and Antonis Michalas. 2021. Blind faith: Privacy-preserving machine learning using function approximation. In 2021 IEEE Symposium on Computers and Communications (ISCC). IEEE, 1–7.
7. Tanveer Khan and Antonis Michalas. 2023. Learning in the Dark: Privacy-Preserving Machine Learning using Function Approximation. (2023).
8. Tanveer Khan, Khoa Nguyen, and Antonis Michalas. 2023. A More Secure Split: Enhancing the Security of Privacy-Preserving Split Learning. In Nordic Conference on Secure IT Systems. Springer, 307–329.
9. Tanveer Khan, Khoa Nguyen, and Antonis Michalas. 2023. Split Ways: Privacy-Preserving Training of Encrypted Data Using Split Learning. In 2023 Workshops of the EDBT/ICDT Joint Conference, EDBT/ICDT-WS 2023, 28 March 2023. CEUR-WS.
10. Tanveer Khan, Khoa Nguyen, Antonis Michalas, and Alexandros Bakas. 2023. Love or Hate? Share or Split? Privacy-Preserving Training Using Split Learning and Homomorphic Encryption. In 2023 20th Annual International Conference on Privacy, Security and Trust (PST). IEEE Computer Society, 1–7.
11. Joon-Woo Lee, HyungChul Kang, Yongwoo Lee, Woosuk Choi, Jieun Eom, Maxim Deryabin, Eunsang Lee, Junghyun Lee, Donghoon Yoo, Young-Sik Kim, et al. 2022. Privacy-preserving machine learning with fully homomorphic encryption for deep neural network. IEEE Access

10 (2022), 30039–30054.

12. Qian Lou, Bo Feng, Geoffrey Charles Fox, and Lei Jiang. 2020. Glyph: Fast and accurately training deep neural networks on encrypted data. Advances in Neural Information Processing Systems 33 (2020), 9193–9202.
13. K Nguyen, T Khan, and A Michalas. 2023. Split Without a Leak: Reducing Privacy Leakage in Split Learning. In 19th EAI International Conference on Security and Privacy in Communication Networks (SecureComm'23). Springer.
14. Alexandros Bakas, Eugene Frimpong, and Antonis Michalas. 2022. Symmetrical Disguise: Realizing Homomorphic Encryption Services from Symmetric Primitives. In International Conference on Security and Privacy in Communication Systems. Springer, 353–370.
15. Christoph Dobraunig, Lorenzo Grassi, Lukas Helminger, Christian Rechberger, Markus Schofnegger, and RomanWalch. 2023. Pasta: a case for hybrid homomorphic encryption. Transaction on Cryptographic Hardware and Embedded Systems 2023 Issue 3 (2023).
16. [Jihoon Cho, Jincheol Ha, Seongkwang Kim, ByeongHak Lee, Joohee Lee, Jooyoung Lee, Dukjae Moon, and Hyojin Yoon. 2021. Transciphering framework for approximate homomorphic encryption. In International Conference on the Theory and Application of Cryptology and Information Security. Springer, 640–669.
17. Jincheol Ha, Seongkwang Kim, Byeonghak Lee, Jooyoung Lee, and Mincheol Son. 2022. Rubato: Noisy Ciphers for Approximate Homomorphic Encryption. In Advances in Cryptology–EUROCRYPT 2022: 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30–June 3, 2022, Proceedings, Part I. Springer, 581–610.
18. Orel Cosseron, Clément Hoffmann, Pierrick Méaux, and François-Xavier Standaert. 2023. Towards Case-Optimized Hybrid Homomorphic Encryption: Featuring the Elisabeth Stream Cipher. In Advances in Cryptology – ASIACRYPT 2022: 28th International Conference on the Theory and Application of Cryptology and Information Security (Taipei, Taiwan). Springer-Verlag, Berlin, Heidelberg, 32–67.
19. Ehsan Hesamifard, Hassan Takabi, Mehdi Ghasemi, and Rebecca N Wright. 2018. Privacy-preserving machine learning as a service. Proc. Priv. Enhancing Technol. 2018, 3 (2018), 123–142.
20. Amartya Sanyal, Matt Kusner, Adria Gascon, and Varun Kanade. 2018. TAPAS: Tricks to accelerate (encrypted) prediction as a service. In International Conference on Machine Learning. PMLR, 4490–4499.
21. Florian Bourse, Michele Minelli, Matthias Minihold, and Pascal Paillier. 2018. Fast homomorphic evaluation of deep discretized neural networks. In Annual International Cryptology Conference. Springer.
22. Sinem Sav, Apostolos Pyrgelis, Juan R Troncoso-Pastoriza, David Froelicher, Jean-Philippe Bossuat, Joao Sa Sousa, and Jean-Pierre Hubaux. 2021. POSEIDON: privacy-preserving federated neural network learning. In 28th Annual Network and Distributed System Security Symposium, NDSS 2021, virtually, February 21-25, 2021. The Internet Society.
23. Ahmad Al Badawi, Chao Jin, Jie Lin, Chan Fook Mun, Sim Jun Jie, Benjamin Hong Meng Tan, Xiao Nan, Khin Mi Mi Aung, and Vijay Ramaseshan Chandrasekhar. 2020. Towards the alexnet moment for homomorphic encryption: Hcnn, the first homomorphic cnn on encrypted data with gpus. IEEE Transactions on Emerging Topics in Computing 9, 3 (2020), 1330–1343.
24. Anne Canteaut, Sergiu Carpov, Caroline Fontaine, Tancrède Lepoint, María Naya-Plasencia, Pascal Paillier, and Renaud Sirdey. 2018. Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression. Journal of Cryptology 31, 3 (2018), 885–916.
25. Pierrick Méaux, Claude Carlet, Anthony Journault, and François-Xavier Standaert. 2019. Improved filter permutators for efficient FHE: Better instances and implementations. In International Conference on Cryptology in India. Springer.