Privacy-Preserving Machine Learning Using Secure Multi-Party Computation

Dr. Anuradha R. More[1], Dr. Shubhangi S. Bhangale[2]
[1]*Design, Analytics and Cyber Security Department, MEER's MIT ACS College, Alandi(D)*
[2]*Science and Computer Science Department, MEER's MIT ACS College, Alandi(D)*
doi.org/10.64643/IJIRT211375

***Abstract*—Machine learning works best when it learns from a lot of data. But the most useful data, such as medical records, bank transactions and personal messages, is also the most private. Often several organizations would like to train a model together, yet none of them is allowed, or willing, to share its raw data. Secure Multi-Party Computation (MPC) offers a way out. It lets many parties compute a result together while keeping each party's input secret. This paper explains, in plain language, how MPC and machine learning can be combined. We describe the main building blocks (secret sharing, garbled circuits, homomorphic encryption), show how common machine learning models such as linear regression, logistic regression and neural networks can be trained and used under MPC, and discuss the cost in speed and accuracy. We also propose a simple hybrid framework that combines MPC with differential privacy and an evaluation plan to test it. Finally, we discuss open challenges and future directions.**

***Index Terms*—privacy-preserving machine learning, secure multi-party computation, secret sharing, federated learning, differential privacy, homomorphic encryption**

I. INTRODUCTION

Imagine three hospitals. Each has records of patients with a certain disease. If they could combine their records, a machine learning model could detect the disease earlier and more accurately. But privacy laws and patient trust say the records must not leave each hospital. So, the hospitals are stuck: the data is valuable, but it cannot be shared.

This situation appears everywhere:

- Banks want to detect fraud across institutions without revealing customer transactions.
- Companies want to measure advertising results together without exposing customer lists.
- Governments and researchers want to study sensitive census or health data without seeing individual records.

Privacy-preserving machine learning (PPML) is the field that tries to solve this. One of its strongest tools is Secure Multi-Party Computation (MPC). The idea is simple to say and surprising in practice: a group of people can compute a function of their private inputs, and learn only the answer, nothing else.

In this paper we look at how machine learning can be done inside MPC. Our goals are:

1. To explain MPC and its link to machine learning in simple words.
2. To review the main methods used to train and run models securely.
3. To discuss the real costs and limits.
4. To propose a practical hybrid design and a plan to evaluate it.

1.1. Contributions

This paper makes four contributions:

1. A plain-language map of how MPC protocols (secret sharing, garbled circuits, homomorphic encryption) fit the needs of machine learning.
2. A side-by-side comparison of well-known private learning systems by number of parties, attacker model and main technique.
3. A hybrid design that joins secret-shared training, secure aggregation and differential privacy, so both the computation and the released model are protected.
4. A clear evaluation plan and an honest list of limits, so other researchers can test the design and judge how far the claims reach.

II. BACKGROUND

2.1 What is Secure Multi-Party Computation?

MPC was first introduced through Andrew Yao's famous "millionaires' problem": two millionaires want to know who is richer without telling each other how much money they have [1]. Later work showed that any computation can, in principle, be done this way [2].

In MPC, there are *n* parties. Each has a private input. Together they run a protocol so that:

- Privacy: nobody learns anything about the others' inputs, except what the final result already reveals.
- Correctness: the output is exactly the right answer, even if some parties try to cheat.

Two common assumptions about attackers are:

- Semi-honest (passive): parties follow the rules but are curious and try to learn extra by looking at what they receive.
- Malicious (active): parties may break the rules, send fake messages or try to ruin the result.

Malicious security is stronger but slower, so most machine learning systems start with the semi-honest model.

2.2 The main building blocks

Secret sharing.

A secret number is split into random pieces called *shares*, and each party gets one piece. Any single share looks like random noise, but together they rebuild the secret [3]. For example, to hide the number 10 between two parties, pick a random number such as 7. Give 7 to Party A and 3 to Party B. Neither share alone says anything, but 7 + 3 = 10. The magic is that parties can *add* shares locally without talking. If Party A holds 7 of a secret 10 and 2 of a secret 5, and Party B holds 3 and 3, then A adds its pieces (7 + 2 = 9) and B adds its pieces (3 + 3 = 6). Together 9 + 6 = 15, which is exactly 10 + 5, and nobody ever saw 10 or 5. So addition is free. Multiplication needs a little communication, usually helped by pre-made random "multiplication triples" [4].

Garbled circuits.

One party turns a computation into an encrypted logic circuit, and the other party evaluates it without learning the inner values [1]. This works well for comparisons and decisions (is this number bigger than that one?).

Homomorphic encryption (HE).

Data is encrypted in a way that still allows calculations on the encrypted values. The result, once decrypted, equals the result of doing the calculation on the plain data [5]. HE uses little communication but a lot of computing power.

Oblivious transfer (OT).

A tool that lets one party pick one item out of several from another party, without the sender learning which item was chosen and without the receiver learning the others. It is the engine behind many MPC protocols.

2.3 Why machine learning is hard to run inside MPC

Machine learning models are mostly built from three kinds of operations:

- Additions and multiplications (matrix products): easy in MPC, especially with secret sharing.
- Comparisons (for example ReLU, max-pooling): harder, because MPC works with numbers and not with "if" statements.
- Non-linear functions (sigmoid, softmax, tanh): hardest, because they involve exponentials and divisions that are expensive to compute on secret values.

Also, MPC works on whole numbers in a finite range, while machine learning uses decimals. So, decimals are turned into fixed-point numbers (for example, storing 3.14 as 314 with an agreed scale), which adds small rounding errors.

2.4 Problem setting and threat model

We study the case where *k* data owners each hold a private dataset and want one shared model trained on all of it. They do not trust each other with raw data. Three computing servers run the secure training, and the data owners send them only secret shares.

Who is protected. Every data owner's raw record, and the model weights while training is still in progress.

Who the attacker is. We assume a semi-honest attacker who follows the protocol but studies everything it sees. At most one of the three servers is corrupt, and no data owner works with it. This is the "honest majority" setting used by ABY3, SecureNN and Falcon [10, 11, 12].

What the attacker can do. Read all messages sent to the corrupt server, and later query the released model many times.
What is out of scope. Servers that break the rules on purpose (malicious security), side-channel attacks on hardware, and poisoned training data from a dishonest owner. Section 7.4 returns to these limits.
What we promise. The corrupt server learns nothing about any single record during training, and the released model leaks only a small, measured amount, set by the differential privacy budget epsilon.

III. RELATED WORK

Researchers have built many systems that join MPC and machine learning. We group them by what they achieve.
Secure inference with encrypted models or inputs.
CryptoNets showed that a neural network can classify encrypted images using homomorphic encryption [6]. Later systems such as GAZELLE and Delphi mixed, HE for linear layers with garbled circuits for non-linear layers, making private prediction much faster [7, 8].

Secure training with secret sharing.
SecureML was one of the first systems to train linear regression, logistic regression and small neural networks where the data owners secretly share their data between two non-colluding servers [9]. ABY3 extended the idea to three servers, which allows faster and more flexible protocols [10]. SecureNN and Falcon designed efficient protocols for the non-linear parts of neural networks and reached practical speeds for training and inference [11, 12].

Frameworks for researchers.
ABY gave a way to switch between arithmetic sharing, boolean sharing and Yao's circuits depending on what is cheapest [4]. CrypTen brought MPC into the familiar PyTorch environment so that machine learning developers can use secure computation without being cryptography experts [13].

Related privacy tools.
Federated learning lets devices train a shared model while keeping raw data local, but the shared updates can still leak information [14, 15]. Secure aggregation uses MPC ideas so the server only sees the *sum* of many updates and not any single one [16]. Differential privacy adds carefully measured noise so that the output reveals very little about any one person [17]. Attacks such as membership inference show why these protections matter [18].

3.1 Comparison of existing systems

The table sums up the systems above. It shows that most work uses either two parties with heavy cryptography, or three parties with lighter secret sharing.

System	Does what	Parties	Attacker model	Main technique
CryptoNets [6]	Inference	Client and server	Semi-honest	Homomorphic encryption
SecureML [9]	Training and inference	2 non-colluding servers	Semi-honest	Secret sharing, garbled circuits
GAZELLE [7]	Inference	2	Semi-honest	Homomorphic encryption, garbled circuits
ABY3 [10]	Training and inference	3 (honest majority)	Semi-honest and malicious versions	Mixed arithmetic, binary and Yao sharing
SecureNN [11]	Training and inference	3	Semi-honest, with a malicious version	Secret sharing
Delphi [8]	Inference	2	Semi-honest	Homomorphic encryption, garbled circuits, secret sharing
Falcon [12]	Training and inference	3 (honest majority)	Malicious	Secret sharing
CrypTen [13]	Training and inference	2 or more	Semi-honest	Secret sharing, with a helper for random values
Secure aggregation [16]	Federated updates	Many clients and a server	Semi-honest, with an active-server version	Masking, secret sharing

This summary is taken from the cited papers; please check details against the originals before quoting them. Our framework in Section 5 follows the three-server design because it gives the best balance of speed and strength for training.

IV. MACHINE LEARNING METHODS UNDER MPC

This section shows how common models are adapted. We keep the explanations short and intuitive.

4.1 Linear regression

Linear regression mostly uses additions and multiplications. Its training step (gradient descent) is a series of matrix products, so it is one of the easiest models to run on secret shares. Each party shares its data, the servers compute the gradients on the shares, and the model weights stay secret until the parties agree to reveal the final model.

4.2 Logistic regression

Logistic regression needs the sigmoid function, which contains an exponential and is expensive in MPC. A practical trick is to replace it by a simple piecewise-linear approximation. For example, output 0 for very small inputs, 1 for very large inputs, and a straight line in between [9]. This costs a little accuracy but saves a lot of time.

4.3 Decision trees and random forests

Decision trees ask a series of questions such as "is age greater than 40?". Each question is a comparison, and the path taken could leak information, so protocols must hide which branch was followed. Garbled circuits and oblivious transfer are good tools here. Training trees securely is more costly than evaluating them.

4.4 Neural networks

Neural networks are the most demanding. A common recipe is:

- Linear layers (fully connected, convolution): use secret sharing or HE.
- ReLU: requires a secure comparison. Methods include garbled circuits or special bit-based protocols [11, 12].
- Max-pooling: replace by average pooling, which is just addition and a scale, or compute the maximum securely.
- Softmax: often left out during private inference because only the largest output is needed. During training it can be approximated.
- Activation functions: replace expensive ones by low-degree polynomials such as x^2, as done in CryptoNets [6].

A useful rule is to design the model to be MPC-friendly from the start: fewer layers, simple activations and smaller sizes. A slightly simpler model that runs ten times faster is often the right choice.

4.5 Federated learning with secure aggregation

In federated learning each participant trains locally and sends model updates. With secure aggregation, each update is masked using secret sharing so the server can only compute the total of all updates [16]. This protects individuals from the server and is far cheaper than training everything inside MPC. Its limit is that the final aggregated model can still leak something, which is why we add differential privacy in the next section.

V. A SIMPLE HYBRID FRAMEWORK

No single tool protects everything. MPC hides data *during* computation. Differential privacy limits what the *output* reveals. Combining them covers both gaps. We propose a lightweight design with four steps.

Step 1: Local preparation.
Each party cleans its data and converts numbers to fixed-point form. Features are scaled to a small range to avoid overflow in secret computation.

Step 2: Secret sharing.
Each party splits its data (or its model update) into shares and sends one to each of three computing servers. Using three servers with at most one corrupt server allows efficient protocols [10].

Step 3: Secure training.
The servers train an MPC-friendly model (for example logistic regression or a small neural network with ReLU and average pooling) directly on the shares.

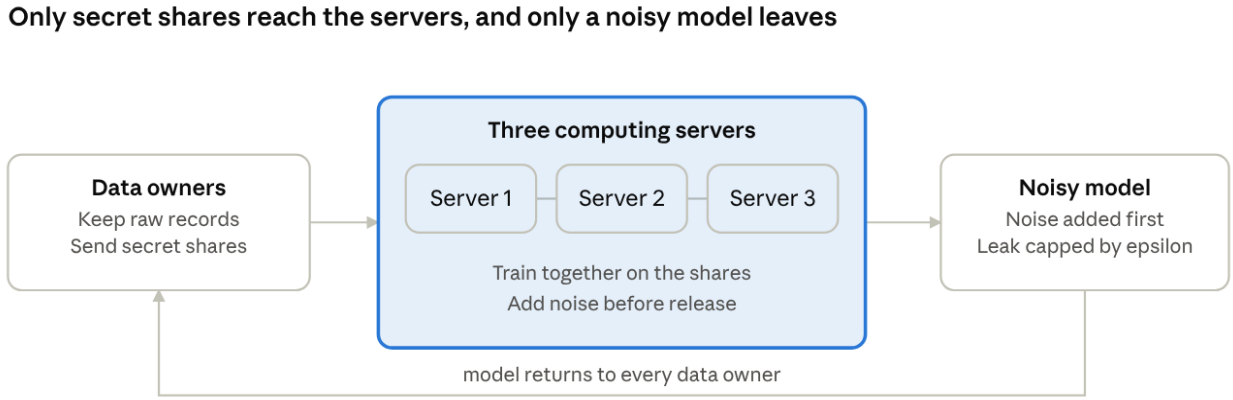
Non-linear parts use approximations or secure comparison.

Step 4: Noise and release.
Before the final model is revealed, the servers add noise inside MPC, following differential privacy. Only the noisy model is revealed to the parties, and the number of queries to the model is limited.

Why this design is sensible:

- Raw data never appears in one place.
- Even the people running the servers cannot see individual records.
- The released model has a measurable privacy guarantee, which blocks many attacks that try to recover training data from a model.

5.1 Architecture



Data owners send only secret shares to three servers. The servers train together and add noise inside MPC, so the model that comes back to the owners is a noisy one.

VI. EVALUATION PLAN

This paper presents a design and review, not a finished experiment, so we do not claim specific numbers. Here is how a study of the framework should be carried out.

Datasets.
Public datasets that are easy to repeat: MNIST (handwritten digits) for images, and a public medical or credit-risk dataset for tabular data.

Models.
Logistic regression, a small multilayer network and a small convolutional network.

Baselines.

1. Normal training with no privacy (best-case accuracy and speed).
2. MPC only.
3. Differential privacy only.
4. The proposed hybrid.

What to measure.

Measure	Why it matters
Accuracy / F1-score	Does privacy hurt the quality of the model?
Training and prediction time	Is it fast enough to use?
Communication (data sent between servers)	Often the real bottleneck in MPC
Privacy budget (epsilon)	How strong is the privacy guarantee?
Attack success rate	Do membership inference or gradient leakage attacks still work?

Test conditions.
Run in a fast local network and in a slower wide-area network, with 3 servers and with more data owners, to see how the system scales.

Expected pattern (hypothesis, not a result).
Based on earlier studies, we expect MPC to give nearly the same accuracy as normal training, with higher time and communication cost, and the added noise to cause a small, controllable drop in accuracy in exchange for stronger privacy.

VII. DISCUSSION

7.1 The trade-off triangle

Private machine learning is a balance between three things: privacy, accuracy and speed. Improving one usually costs another.

- More privacy (more noise, stronger attackers assumed) lowers accuracy or slows things down.
- More accuracy (exact functions, big models) raises cost.
- More speed (approximations, smaller models) may reduce accuracy.

A good system lets the user choose their own balance for their situation.

7.2 What MPC does *not* protect

MPC is not magic. It protects the computation, but:

- The final answer may still reveal something. A very accurate model can leak facts about its training data.
- If parties collude, more than the system allows, privacy can break.
- Poor input data (wrong labels, biased samples) stays a problem; hidden data is also harder to check for errors or bias.

7.3 Practical challenges

- Communication cost: servers must exchange many messages, so slow networks hurt.
- Scale: training large modern models under MPC is still far slower than normal training.
- Usability: engineers need tools that feel like normal machine learning libraries; CrypTen is a step in that direction [13].
- Malicious security: stronger protection against cheating costs several times more.
- Law and trust: organizations need clear rules about who runs servers and who may see the result.

7.4 Limitations of this study

This paper proposes and reviews; it does not yet prove the design works at scale. Readers should keep five limits in mind.

- No measured results. The evaluation in Section 6 is a plan. The expected pattern is a hypothesis drawn from earlier studies.
- Honest-majority assumption. If two of the three servers cooperate, privacy fails. Choosing servers run by different organizations lowers this risk but does not remove it.
- Semi-honest attacker only. A server that cheats on purpose could corrupt the model. Malicious-secure protocols such as Falcon address this at a higher cost [12].
- Small models. The design targets logistic regression and small neural networks. Large modern models remain out of reach for now.
- Data quality and poisoning. Secret sharing hides data, so a dishonest owner can also hide bad data. Extra checks would be needed.

Future work should run the planned experiments and report accuracy, time, communication and attack results together.

VIII. FUTURE DIRECTIONS

1. Faster non-linear functions such as softmax and GELU, which large modern networks depend on.
2. Secure training of transformer models and large language models, which today is very expensive.
3. Hardware help, using GPUs or trusted hardware to speed up secret-shared operations.
4. Automatic privacy tuning, where a tool picks the best balance between noise, approximation and speed.
5. Verifiable computation, so parties can *check* that the training was done correctly without seeing the data.
6. Real-world pilots in healthcare, finance and public statistics to learn what matters outside the lab.

IX. CONCLUSION

Secure multi-party computation gives machine learning something it badly needs: a way to learn from data that nobody is allowed to share. Methods such as secret sharing, garbled circuits and homomorphic encryption now make it possible to train and use real models with strong privacy. The price is extra time and communication, and some care in choosing simple, MPC-friendly models. Combining MPC with differential privacy and secure aggregation protects both the process and the final output. Although large-scale models are still out of easy reach, rapid progress in protocols, tools and hardware suggests that private collaborative learning will soon be a normal part of how organizations work together.

REFERENCES

[1] A. C. Yao, “How to generate and exchange secrets,” in *Proc. 27th IEEE Symp. Foundations Comput. Sci. (FOCS)*, 1986.

[2] O. Goldreich, S. Micali, and A. Wigderson, “How to play any mental game,” in *Proc. 19th ACM Symp. Theory Comput. (STOC)*, 1987.

[3] A. Shamir, “How to share a secret,” *Commun. ACM*, vol. 22, no. 11, 1979.

[4] D. Demmler, T. Schneider, and M. Zohner, “ABY: A framework for efficient mixed-protocol secure two-party computation,” in *Proc. Network and Distributed System Security Symp. (NDSS)*, 2015.

[5] C. Gentry, “Fully homomorphic encryption using ideal lattices,” in *Proc. 41st ACM Symp. Theory Comput. (STOC)*, 2009.

[6] R. Gilad-Bachrach *et al*., “CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy,” in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2016.

[7] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, “GAZELLE: A low latency framework for secure neural network inference,” in *Proc. USENIX Security Symp.*, 2018.

[8] P. Mishra *et al*., “Delphi: A cryptographic inference service for neural networks,” in *Proc. USENIX Security Symp.*, 2020.

[9] P. Mohassel and Y. Zhang, “SecureML: A system for scalable privacy-preserving machine learning,” in *Proc. IEEE Symp. Security Privacy*, 2017.

[10] P. Mohassel and P. Rindal, “ABY3: A mixed protocol framework for machine learning,” in *Proc. ACM Conf. Comput. Commun. Security (CCS)*, 2018.

[11] S. Wagh, D. Gupta, and N. Chandran, “SecureNN: 3-party secure computation for neural network training,” *Proc. Privacy Enhancing Technol. (PoPETs)*, 2019.

[12] S. Wagh *et al*., “Falcon: Honest-majority maliciously secure framework for private deep learning,” *Proc. Privacy Enhancing Technol. (PoPETs)*, 2021.

[13] B. Knott *et al*., “CrypTen: Secure multi-party computation meets machine learning,” in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2021.

[14] B. McMahan *et al*., “Communication-efficient learning of deep networks from decentralized data,” in *Proc. Int. Conf. Artif. Intell. Stat. (AISTATS)*, 2017.

[15] L. Zhu, Z. Liu, and S. Han, “Deep leakage from gradients,” in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2019.

[16] K. Bonawitz *et al*., “Practical secure aggregation for privacy-preserving machine learning,” in *Proc. ACM Conf. Comput. Commun. Security (CCS)*, 2017.

[17] C. Dwork, F. McSherry, K. Nissim, and A. Smith, “Calibrating noise to sensitivity in private data analysis,” in *Proc. Theory Cryptography Conf. (TCC)*, 2006.

[18] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, “Membership inference attacks against machine learning models,” in *Proc. IEEE Symp. Security Privacy*, 2017.