

Privacy-Preserving Machine Learning Using Fully Homomorphic Encryption

Introduction

Machine learning can discover useful patterns in data. Many systems today depend on this process. Health care, finance, mobile services, and public institutions use machine learning to make decisions or produce predictions. These systems often depend on sensitive data. When this data is sent to a server for training or prediction, it can expose private information. Fully homomorphic encryption, also called FHE, is a cryptographic tool that can protect this information. FHE lets a server run computations on encrypted data. The server never learns the real values. Only the data owner can decrypt the result. This makes FHE a strong option for privacy-preserving machine learning.

This paper reviews how FHE began and how it grew into a practical tool. It explains how early ideas became faster schemes. It also shows how FHE can support encrypted training and encrypted inference. This review examines software libraries that make these methods usable today. It then evaluates the current limits of the field. Last, it describes how these ideas may reach real-world deployment. The goal is to give a clear picture of the state of the art in privacy-preserving machine learning with FHE.

Functionality and Purpose

The main purpose of FHE is to allow computations on encrypted data. Traditional encryption only protects data when it is stored or sent. Once a system needs to use the data, it must decrypt it. This

step exposes sensitive values. FHE avoids this exposure. With FHE, a user encrypts their data and sends the ciphertext to a server. The server performs additions, multiplications, or full programs on this ciphertext. The server returns the encrypted result. Only the user can decrypt and see the output. This process protects data during use. It does not leak information to the party doing the computation. In the context of machine learning, this means a server can train a model or run an inference step without ever learning anything about the user's data.

In privacy-preserving machine learning, FHE supports two tasks. The first is encrypted training. This means a model learns from encrypted samples. The second is encrypted inference. This means the model receives encrypted input and sends back an encrypted prediction. Both tasks let the system operate without exposing sensitive data. This is useful for medical records, financial statements, location data, and many other forms of private information.

Foundational Work

The modern study of FHE began with the work of Craig Gentry (2009). Before this, researchers had partial schemes. These partial schemes could perform either encrypted addition or encrypted multiplication, but not both on the same ciphertext. Gentry showed the first complete solution. His scheme used ideal lattices to support an unlimited number of operations on encrypted data (Gentry, 2009). The key idea was a method called bootstrapping. Bootstrapping refreshes a noisy ciphertext. It allows the system to run more operations without losing correctness. This idea made FHE possible. It also created a new line of research in cryptography.

Gentry's scheme proved that FHE was possible. However, it was too slow for practical use. Even simple operations required large memory and long computation time. But the core idea shaped the path forward. Researchers then focused on improving speed, reducing memory, and lowering

parameter sizes. This first work is still the base of every modern scheme. It remains one of the most important results in cryptography.

Major Developments in FHE

After Gentry's work, several new schemes improved speed and efficiency. The BGV scheme introduced better methods for handling noise growth and modular switching. This made the system more practical for layered computations. The BFV scheme improved integer arithmetic. The CKKS scheme allowed approximate arithmetic for real numbers. CKKS is widely used in machine learning because it supports vectorized operations on floating-point values. These advances helped create a full ecosystem of practical FHE tools.

Researchers also worked on faster bootstrapping. Early bootstrapping was extremely slow. Many operations required minutes or hours. Later methods reduced this time by several orders of magnitude. Today, some schemes support bootstrapping in fractions of a second. These improvements allow deeper machine learning models and more flexible applications.

Another major development was packing. Packing lets many values fit inside a single ciphertext. A single operation can then affect an entire vector. Machine learning benefits from this because models often depend on matrix and vector operations. Packing reduces cost and memory for these tasks. Together, these developments turned FHE from a theory result into a tool that developers can use for real applications.

FHE in Machine Learning

Machine learning uses large amounts of data. It also uses operations that fit well with the kinds of arithmetic supported by FHE. For example, linear models, logistic regression, and neural network layers rely on addition and multiplication. This makes them suitable for encrypted computation.

Many early studies explored encrypted training for simple models. One example is secure logistic regression applied to medical records (Lee et al., 2018). This study used homomorphic encryption to train on sensitive health data. The researchers showed that the model accuracy stayed high while patient privacy remained protected. Their work demonstrated that private machine learning was possible outside of theory.

Encrypted inference is now the most active area of FHE-based machine learning. Inference requires fewer operations than training. This makes it more practical today. A user can encrypt their input. A server can run the model on the encrypted input. The server returns an encrypted prediction. Only the user can decrypt it. This protects sensitive queries. It also protects the model from exposure. As a result, FHE can support two-sided privacy.

There are also hybrid designs. Some systems train in the clear but use FHE for private inference. Others split training into stages so that only some parts run in encrypted form. These methods reduce cost and still provide privacy where needed. As FHE speeds improve, more complex neural networks become feasible. This includes convolutional networks, transformers, and even smaller generative models.

Software Tooling and Practical Systems

Modern FHE work depends on strong software libraries. Microsoft SEAL is one of the most widely used. It implements the BFV and CKKS schemes. SEAL is designed for developers. It offers clear functions for encryption, evaluation, and decryption. It also includes tools for parameter selection and memory management (Chen et al., 2017). Many machine learning systems use SEAL as the base layer for encrypted arithmetic.

OpenFHE is another major library. It supports multiple schemes, including CKKS, BFV, BGV, and others. It also includes advanced features like bootstrapping, packing, and optimized

arithmetic for GPU and CPU platforms. These libraries make FHE more accessible. They also reduce the burden of handling complex math details. Without these tools, private machine learning would be much harder to build.

Some companies also provide cloud services based on FHE. These services allow encrypted inference for custom models. While these systems are still early, they show growing interest in commercial use. They also show that FHE is moving from research labs to public deployment.

Current State of the Art

Today's state of the art shows strong improvement in speed, memory use, and model support. Many encrypted inference tasks now run in seconds rather than minutes. Vectorization and packing support batch computation. Bootstrapping is far faster than before. Approximate arithmetic from CKKS allows efficient support for neural networks. Despite these gains, FHE still cannot match the speed of normal machine learning. But the gap is shrinking. This progress makes real systems more realistic.

Security guarantees also remain strong. FHE schemes rely on lattice problems that are believed to be resistant to quantum attacks. This means FHE can support long-term privacy. It also means that private machine learning systems built with FHE can remain secure even as computing power increases.

Challenges and Future Work

FHE still faces major challenges. Computation time is slower than non-encrypted methods. Memory use is high. Parameter selection is difficult. These parameters affect both security and speed. A wrong choice can weaken the system or make it too slow to use. Many users rely on

defaults from libraries. This is not always ideal. Better tools for automatic parameter selection would help.

Training large models under FHE is still very hard. Neural networks with many layers require heavy computation. Bootstrapping adds cost. As a result, encrypted training is uncommon in production. Future work may reduce cost through new math tricks, new hardware support, or hybrid methods. Another challenge is noise growth. Every operation increases noise. If noise becomes too large, the ciphertext breaks. Managing noise while keeping models accurate is an ongoing research problem.

Better integration with GPUs and specialized hardware may solve many performance issues. Some early systems show strong improvements from parallel computation. Future research may develop hardware built specifically for homomorphic operations. This could push the field close to real-world use.

Real-World Use and Outlook

Some systems already use FHE for private inference. Financial firms test encrypted credit scoring. Health groups test private diagnosis tools. These systems run only small models today, but they show clear value. Larger systems will need more speed and better memory use. They will also need strong support tools that help developers manage parameters and performance. As these improvements arrive, privacy-preserving machine learning may become common in cloud services. It could protect personal data in many settings. It could also support fair and safe AI use by limiting data exposure.

Conclusion

Fully homomorphic encryption has moved from theory to practice in the past decade. It began with Gentry's breakthrough work (2009). Since then, faster schemes and better libraries have made it suitable for machine learning tasks. FHE now allows encrypted training for simple models and encrypted inference for more complex ones. Software tools like Microsoft SEAL and OpenFHE have made these ideas reachable for developers. The field has strong security, but it still struggles with speed, memory, and parameter choices. These limits are active areas of research. Continued progress may lead to private AI systems that protect sensitive data without slowing down real-world use. This vision shows the promise of privacy-preserving machine learning and the role that FHE can play in the future.

References

- Craig Gentry. *Fully Homomorphic Encryption Using Ideal Lattices*. STOC 2009. <https://doi.org/10.1145/1536414.1536440>
- Hao Chen et al. *SEAL: A Homomorphic Encryption Library*. Microsoft Research, 2017. <https://www.microsoft.com/en-us/research/project/microsoft-seal/Links to an external site.>
- Jae Kyu Lee et al. *Secure Logistic Regression Based on Homomorphic Encryption: Design and Evaluation*. JMIR Med Inform 2018. <https://pmc.ncbi.nlm.nih.gov/articles/PMC5930176/>