Multi-Party Computation Meets Machine Learning

Ref: CrypTen: Secure Multi-Party Computation Meets Machine Learning, 2021, https://arxiv.org/abs/2109.00984

Secure multi-party computationallows parties to collaboratively perform computations on their combined data sets without revealing the data they possess to each other.

This capability of secure MPC has the potential to unlock a variety of machine-learning applications that are currently infeasible because of data privacy concerns.

For example, secure MPC can allow medical research institutions to jointly train better diagnostic models without having to share their sensitive patient data or allow social scientists to analyze gender wage gap statistics without companies having to share sensitive salary data .

To foster the adoption of secure MPC techniques in machine learning,

CRYPTEN: a flexible software framework that aims to make modern secure MPC techniques accessible to machine- learning researchers and developers without a background in cryptography.

Provides a comprehensive tensor-computation library in which all computations are performed via secure MPC.

Background

Shamir Secret Sharing

Ref: Secure Multiparty Computation (MPC), Yehuda Lindell, https://dl.acm.org/doi/10.1145/3387108 [Section 4.1]

Computing Addition and Multiplication

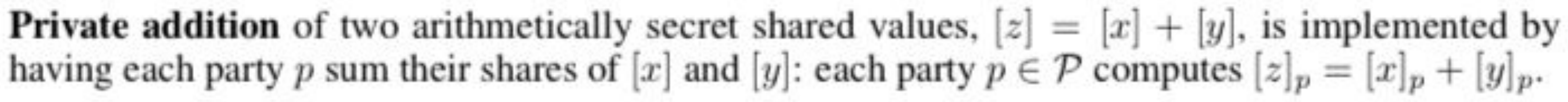
Ref: Secure Multiparty Computation (MPC), Yehuda Lindell, https://dl.acm.org/doi/10.1145/3387108 [Section 4.1]

[Lagrange Interpolation Formula

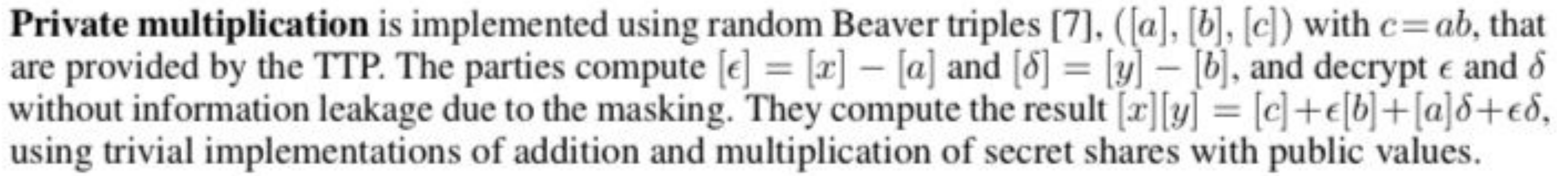
Given few real values x_1, x_2, x_3, …, x_n and y_1, y_2, y_3, …, y_n and there will be a polynomial P with real coefficients satisfying the conditions $P(x_i) = y_i$, $\forall i = \{1, 2, 3, \ldots, n\}$ and degree of polynomial P must be less than the count of real values i.e., degree(P) < n**]**

Ref link: https://youtu.be/_mDlLKgiFDY

Secure Computation



Private addition of two arithmetically secret shared values, $[z] = [x] + [y]$, is implemented by having each party p sum their shares of $[x]$ and $[y]$: each party $p \in \mathcal{P}$ computes $[z]_p = [x]_p + [y]_p$.



Private multiplication is implemented using random Beaver triples [7], $([a], [b], [c])$ with $c = ab$, that are provided by the TTP. The parties compute $[\epsilon] = [x] - [a]$ and $[\delta] = [y] - [b]$, and decrypt ϵ and δ without information leakage due to the masking. They compute the result $[x][y] = [c] + \epsilon[b] + [a]\delta + \epsilon\delta$, using trivial implementations of addition and multiplication of secret shares with public values.

Ref 7 of the paper: D. Beaver. Efficient multiparty protocols using circuit randomization. In *Annual International Cryptology Conference*, pages 420–432. Springer, 1991.

Linear functions are trivially implemented as combinations of private addition and multiplication. This allows CRYPTEN to compute dot products, outer products, matrix products, and convolutions.

Non-linear functions are implemented using standard approximations that only require private addition and multiplication. Specifically, CRYPTEN evaluates exponentials using a limit approximation,

CRYPTEN

Theory : https://arxiv.org/abs/2109.00984

Implementation/ Code:

https://github.com/facebookresearch/CrypTen/blob/main/tutorials/Introduction.ipynb

Sample Code:

https://github.com/facebookresearch/CrypTen/tree/main/examples

https://github.com/facebookresearch/CrypTen/blob/main/examples/mpc_linear_svm/mpc_linear_svm.py