**Benchmarking CrypTen**

DS7406: MLSys Final Paper

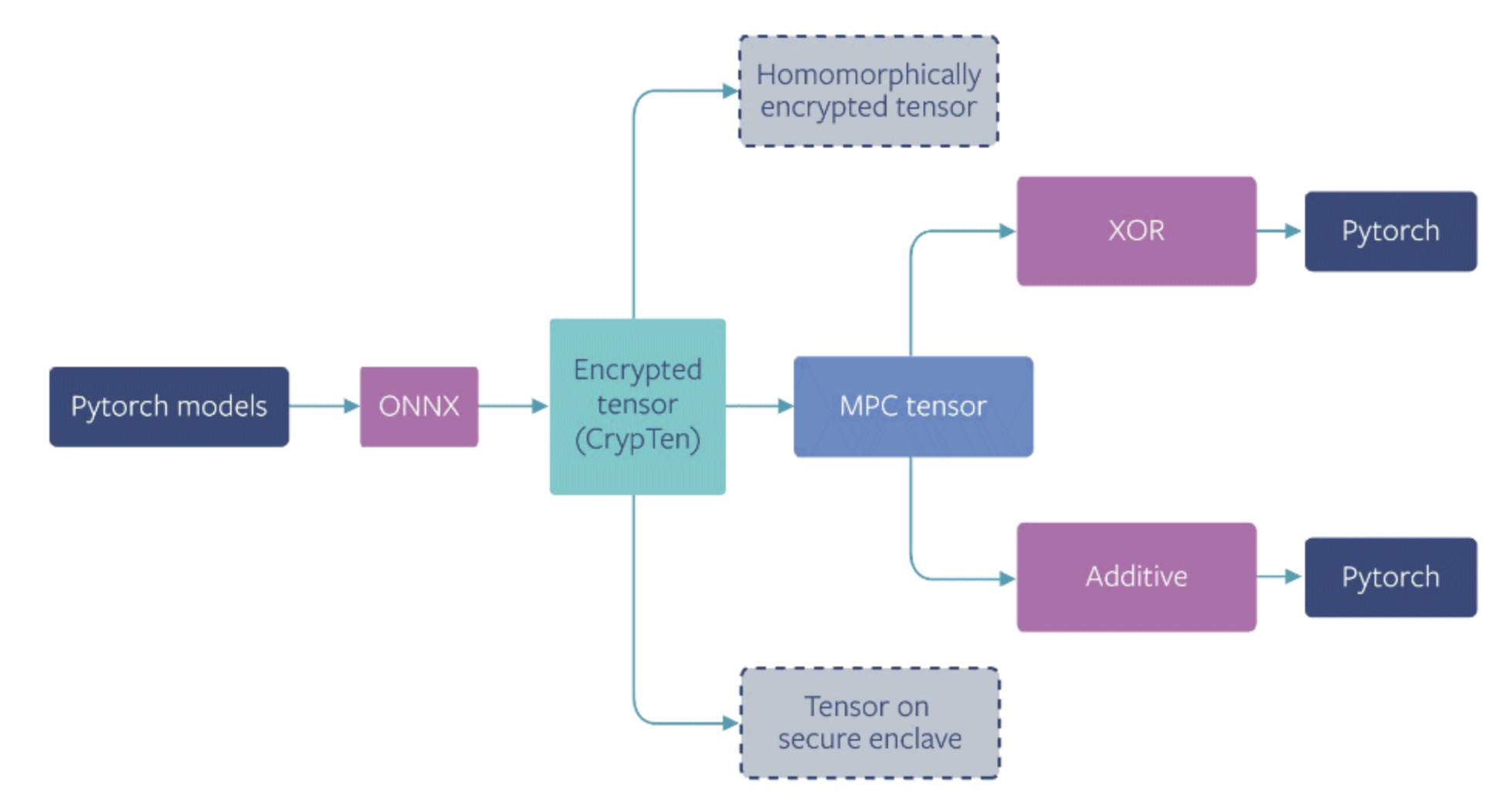
Team 1: Saurav Sengupta, Zachary Jacokes

1. **Introduction**

The advent of cloud computing has enabled great advances in machine learning applications in a wide range of domains. Entities and companies without the requisite computing infrastructure can simply outsource their computing needs to any one of several cloud computing providers in order to employ the latest deep learning techniques for inference. However, this naturally results in greater risk of personal data exposure, so an emphasis on data privacy has become more prevalent alongside this rise in cloud computing. This is especially true in healthcare settings, where personal information is legally protected under Health Insurance Portability and Accountability Act (HIPAA) regulations. Data privacy practices have largely centered around either anonymization and encryption.

Historically, much of the confidentiality-related machine learning work in healthcare settings has focused on anonymization pipelines, where sensitive patient data is systematically assessed for potential identifiers and then scrubbed of these before being analyzed. This avenue is unappealing for two reasons: the first is that it is a computationally expensive and inefficient task, and the second is that there is an uncomfortably high chance for errors, especially in cases where the anonymization process is not fully complete. In real-world settings with patient privacy at stake, these risks are unacceptable and thus a more effective solution is necessary. Encryption is the more modern alternative to full anonymization, and recent research has yielded promising results about its viability in managing healthcare data inference.

This paper aims to assess how effectively a new encryption software framework, CrypTen, can compete with other algorithms in this space[1]. CrypTen is an open-source machine learning and encryption software structure built on top of the PyTorch framework. It is designed to enable deep-learning analyses on sensitive datasets in as simple a manner as possible. The encryption is performed on the front-end, which means that both training and inference are performed on encrypted data. The main appeals of CrypTen include its simple integration with and implementation in PyTorch, in addition to its relative accessibility for those not familiar with cryptography. CrypTen is a relatively new development, and as such comprehensive benchmarking has not been reproduced outside of the original CrypTen whitepaper. This is the motivation behind the current paper and the research gap we aim to address.



**Figure 1:** Full CrypTen pipeline as described by the CrypTen whitepaper (ref: <https://crypten.ai/>)

In this paper, we assess the inference speed and computational efficiency of CrypTen for both an image classification task and text classification task. The image classification task is performed on the MNIST dataset, and for text classification we use a dataset of 200,000 medical abstracts to approximate what a classification algorithm might encounter in a dataset of patient notes. In order to assess the capabilities of CrypTen, we first perform these analyses on unencrypted data using vanilla PyTorch, and then again using the CrypTen framework while holding the dataset, classification task, and deep learning algorithm constant.

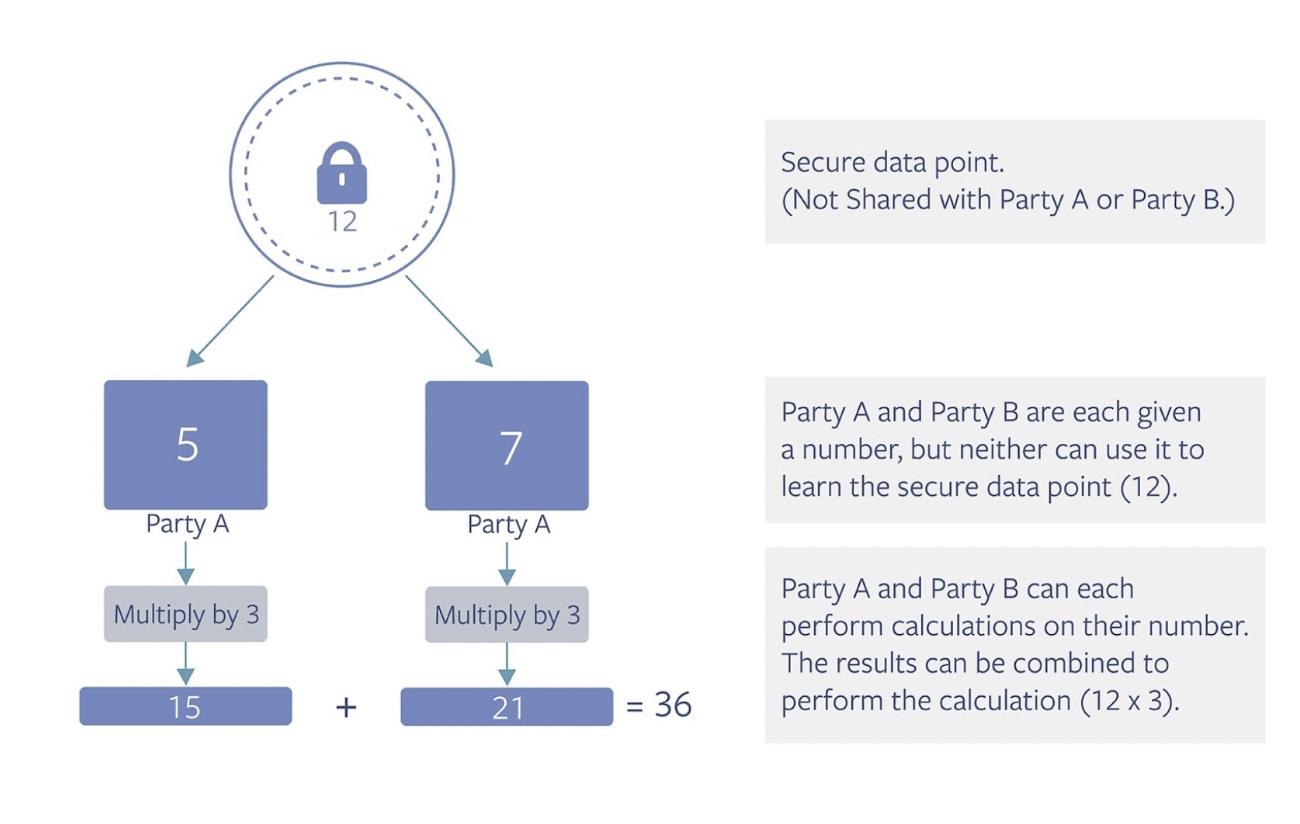
1. **Related Work**

***Homomorphic Encryption***

Homomorphic encryption refers to a collection of protocols through which encrypted data can be manipulated mathematically and return the same result as non-encrypted data. Fully homomorphic encryption is considered the gold standard of encryption because it theoretically allows for an infinite number of mathematical operations of any type on encrypted data. While fully homomorphic encryption would be technically unparalleled in terms of threat resistance, it is mostly infeasible to implement in practical situations due to massive computational costs. The first fully homomorphic encryption algorithm was developed in 2009[12], and while efficiency improvements are still being developed today, we are nowhere near feasibility as it pertains to fully homomorphic encryption.

Partially homomorphic encryption is a far more practical and widely-used encryption solution. It allows for secure computations on encrypted data, but with specific limitations depending on which algorithm is used. In some cases only specific types of operations, including one of either addition or multiplication (not both), can be performed an infinite number of times[13][16]. Other partially homomorphic encryption algorithms can support multiple types of operations, but only a limited number of times[14][15]. CrypTen uses two synchronous partially homomorphic encryption algorithms that fall into the former category to perform most of the computations necessary for deep learning, detailed later in this section.

***Secure Multi-party Computation***



**Figure 2:** Multi-party computation schematic (ref: <https://crypten.ai/>)

Secure multi-party computation (MPC) works by distributing pieces of data across multiple parties such that no individual party has access to the full dataset or solution. This effectively secures computing operations without exposing or moving the data. In a typical scenario, the parties involved in the computation endeavor to calculate the output of a specific function involving all of their private data without exposing it to each other. Figure 2 provides an example for how this would work in practice, where a solution is reached without individual entities revealing their data to each other.

| **Protocol** | **MNIST** | | **ImageNet** | | **Malaria** | |
| --- | --- | --- | --- | --- | --- | --- |
|  | runtime (s) | comm (MB) | runtime (s) | comm (MB) | runtime (s) | comm (MB) |
| Falcon8 | 0.047 | 0.74 | 1.81 | 19.21 | - | - |
| CryptGPU9 | 0.38 | 3.00 | 1.52 | 240 | - | - |
| Gazelle7 | 0.81 | 70.0 | - | - | - | - |
| PySyft10 | 0.66 | 1.2 | - | - | 8.26 | 38.56 |
| TF-Trusted10 | 0.12 | - | - | - | 0.14 | - |
| CrypTen1 | 0.035 | 0.35 | - | - | 0.25 | 8.29 |

**Table 1:** Comparison of other encryption framework performance compared to CrypTen

***Operations in CrypTen***

CrypTen works by implementing secure MPC on mathematical operations. Imagine parties get shares of x and y as [x] and [y]. CrypTen performs the following mathematical operations:

**Private addition**: each party p ∈ P computes [z]p = [x]p + [y]p such that [x]p + [y]p = x + y.

**Private multiplication**: using Beaver triples, ([a], [b], [c]) with c=ab, the parties compute [𝜺] = [x] − [a] and [δ] = [y] − [b] and then compute the result [x][y] = [c]+[b]+[a]δ+𝜺δ.

**Linear operations,** which can be implemented as combinations of private multiplication and private addition.

**Non-Linear functions** like Sigmoid, ReLU etc. are implemented using approximations of additions and multiplications.

Since GPUs are optimized for floating point, operations are done by breaking integer operations into 64-bit floating point operations.

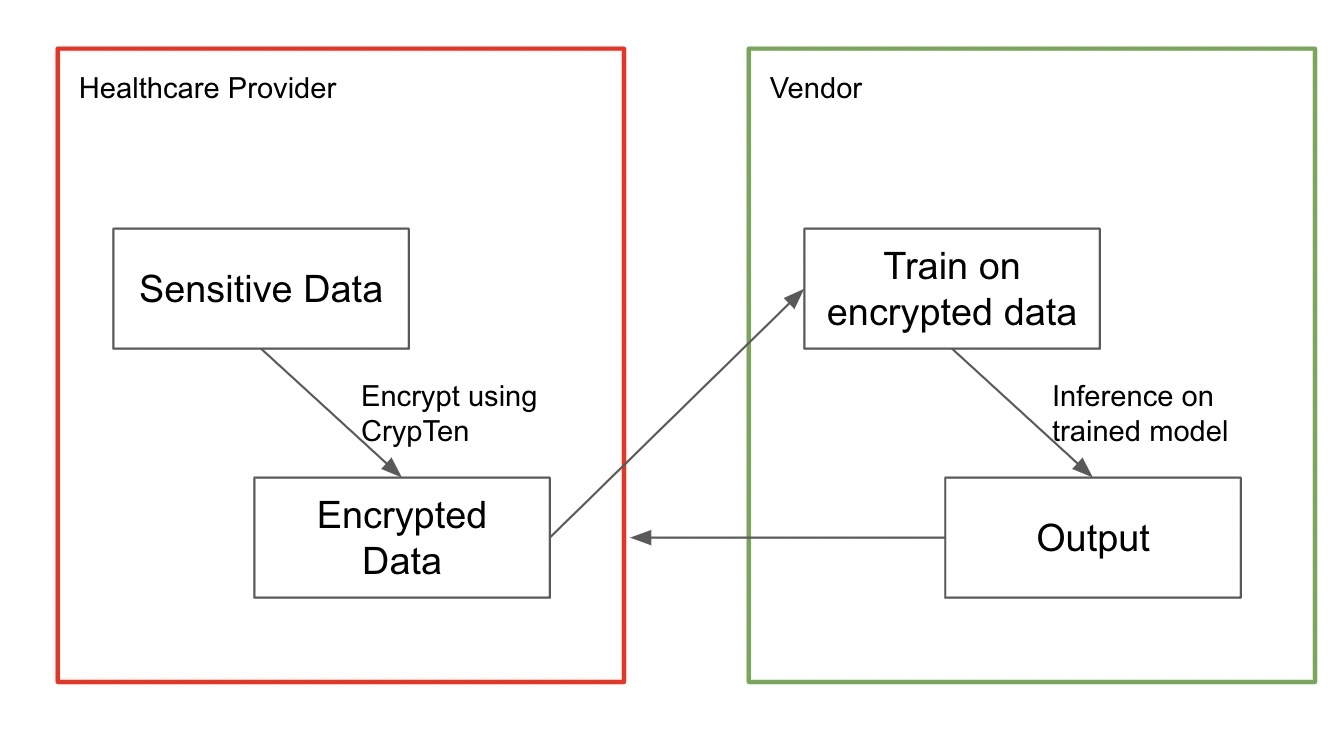
1. **Datasets**

We benchmark CrypTen using two datasets to evaluate the performance of CrypTen on both text and image classification: the MNIST handwritten digit dataset for image classification and 200,000 medical paper abstracts for text classification[5][6]. The MNIST dataset consists of 70,000 images of handwritten digits and is based on data from the American Census Bureau. The images are normalized to be 28 x 28 pixels and is a common image classification benchmarking dataset.

The medical paper abstracts dataset contains 200,000 abstracts, each sentence of which is split into either Objectives, Methods, Results, or Conclusions. The purpose of including this dataset in the analysis is to approximate the text that may be found in the notes of patient medical records, which would be a common use for encrypted inference in healthcare. Sensitive information is contained in patient notes, but they are also a rich source of potential information pertaining to disease identification and treatment routes. Accuracy and efficiency in text classification is a valuable use-case for CrypTen in medical settings.

These datasets were chosen because their large size allows for real world use case generalization to test the performance and scalability of training models using CrypTen. Since benchmarking CrypTen for practical use is the goal of this study, we aim to achieve this goal by imitating real world use of secure multi-party computation and analyzing the datasets in the sense that they would belong to different parties that do not want to expose their private data to the other.

1. **Methodology**



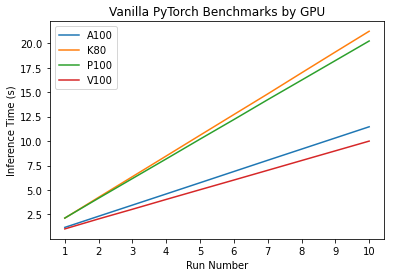
**Figure 3:** Proposed encryption pipeline

The following steps could represent a typical implementation of CrypTen for healthcare machine learning inference:

1. The client, in this case likely the healthcare provider, after reaching out to an encryption service provider, would be sent an encryption key with which to encrypt their sensitive data.
2. The encrypted data would then be sent to the encryption service provider, whereupon the provider would perform any inference task the client is interested in seeing on their data. This inference would be performed on the encrypted data using CrypTen.
3. The encryption service provider would send the encrypted results back to the client.
4. The client would decrypt the results using the same encryption key they initially used to encrypt the data.

This pipeline uniquely allows for private data sharing between clients and providers using cloud computing, thus removing the need for anonymization and potential for human error.

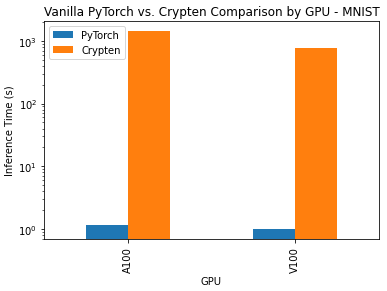
1. **Results**



**Figure 4:** Initial inference time benchmark using vanilla PyTorch on MNIST dataset, separated by GPU.

***MNIST Dataset***

The first step in this analysis was to generate a baseline calibration for inference time using vanilla PyTorch. This was done using a simple CNN architecture (two 3x3 convolutional layers, 2 dropout layers, training batch size = 64). Figure 4 shows the results across 4 GPUs: NVIDIA A100 Tensor Core, NVIDIA Tesla K80, NVIDIA Tesla P100, and NVIDIA V100 Tensor Core. The average inference time (in seconds) per epoch was as follows: A100 = 1.1462, K80 = 2.1233, P100 = 2.0235, V100 = 0.9995. As expected, the newer models (A100, V100) perform considerably faster than the older models (K80, P100). This analysis focuses on inference time for the A100 and V100 GPUs to observe how encrypted inference performs on the latest hardware.



**Figure 5:** Comparison of PyTorch vs. CrypTen on two GPUs using the MNIST dataset.

The next step in the analysis was to compare the performance of vanilla PyTorch and CrypTen on the MNIST dataset. This comparison was performed using the same network architecture and same GPUs; the results are shown in Figure 5. Average inference time for CrypTen was 1426.017 seconds on the A100 and 770.035 seconds on the V100. This is an interesting and surprising result given that the A100 is actually the newer of the GPU hardware designs, so we would have expected the V100 to perform better. We saw no accuracy drop-off from the vanilla PyTorch implementation to the encrypted model. Additionally, it is important to note that while the vanilla PyTorch implementation resulted in minimal differences between the two GPUs, the CrypTen implementation resulted in massive inference time differences: the V100 was almost twice as fast as the A100.

***200,000 Medical Paper Abstract Dataset***

For the text classification task on the medical abstracts dataset, we first attempted to implement a Long Short-Term Memory (LSTM) network. Though this algorithm was able to be successfully employed on vanilla PyTorch, the transition to CrypTen was not successful. We attempted to implement several LSTM architectures, but we learned that CrypTen does not yet support LSTM networks. We shifted focus away from LSTM and toward a one dimensional CNN, which can be used to classify text, but again we experienced issues getting the encrypted architecture to run. We are continuing to work towards getting this dataset classified in an encrypted architecture.

1. **Conclusion**

From the results we were able to generate, we can conclude definitively that CrypTen is far slower than vanilla PyTorch. Importantly, we found that CrypTen appears to have been optimized for the V100 generation of GPUs, evidenced by the fact that our experiment on the A100 GPU resulted in far slower inference time. Though the difference was small, the vanilla PyTorch implementation also performed better on the V100, which we ascribe to configuration issues on Rivanna.

***Limitations***

We were unable to run encrypted models on the K80 and P100 GPUs, likely due to some hardware and versioning incompatibility since these older GPUs were developed before CrypTen. The issues we experienced getting LSTM up and running were to some extent caused by a conversion error with Open Neural Network Exchange (ONNX), which after some digging we found does not support the specific form of text classification we attempted to implement.

***Future Work***

We plan to implement the text classification problem differently in future analyses. The most logical approach is to narrow the focus of the classification task from a multi-class problem to a two-class problem; we believe this could be the cause of many of the errors we encountered.

Another way we could approach this problem is through the use of transformer architectures, which have been shown to work on CrypTen[1] and are effective techniques for text classification.

We also plan to assess how CrypTen inference time is affected by heavy parallelization across nodes and GPUs. We expect inference time to decrease linearly with additional accelerators, though the quirks present in Rivanna may lead to unexpected results.

1. **References:**

[1] Knott, B., Venkataraman, S., Hannun, A., Sengupta, S., Ibrahim, M., & van der Maaten, L. (2021). Crypten: Secure multi-party computation meets machine learning. *Advances in Neural Information Processing Systems*, *34*, 4961-4973.

[2] Bos, J. W., Lauter, K., & Naehrig, M. (2014). Private predictive analysis on encrypted medical data. *Journal of biomedical informatics*, *50*, 234-243.

[3] Wang, F., Zhu, H., Lu, R., Zheng, Y., & Li, H. (2020). Achieve efficient and privacy-preserving disease risk assessment over multi-outsourced vertical datasets. *IEEE Transactions on Dependable and Secure Computing*.

[4] Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, Ronald Summers, ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases, IEEE CVPR, pp. 3462-3471, 2017

[5] https://www.tensorflow.org/datasets/catalog/mnist

[6] https://www.kaggle.com/datasets/anshulmehtakaggl/200000-abstracts-for-seq-sentence-classification

[7] https://github.com/facebookresearch/CrypTen/blob/main/tutorials/Tutorial\_2\_Inside\_CrypTensors.ipynb

[8] Chiraag Juvekar, Vinod Vaikuntanathan, Anantha Chandrakasan. Gazelle: A Low Latency Framework for Secure Neural Network Inference. In: Proceedings of the USENIX Security Symposium, 2018.

[9] Sijun Tan, Brian Knott, Yuan Tian, David Wu. CryptGPU: Fast Privacy-Preserving Machine Learning on the GPU. In: Proceedings of the IEEE Symposium on Security and Privacy, 2021.

[10] Veneta Haralampieva, Daniel Rueckert, Jonathan Passerat-Palmbach. A Systematic Comparison of Encrypted Machine Learning Solutions for Image Classification. In: Proceedings of the PPMLP, 2020.

[11] Morten Dahl, Jason Mancuso, Yann Dupis, Ben Decoste, Morgan Giraud, Ian Livingstone, Justin Patriquin, Gavin Uhma. Private Machine Learning in TensorFlow using Secure Computation. arXiv, Oct. 23, 2018. Accessed: Oct. 15, 2022.

[12] Craig Gentry. A Fully Homomorphic Encryption Scheme. 2009.

[13] Shaft G, Silvio M (1982) Probabilistic encryption & how to play mental poker keeping secret all partial information. In: Proceedings of the annual ACM symposium on theory of computing, pp 365–377.

[14] Ronald LR, Adi S, Leonard MA () A method for obtaining digital signatures and public-key cryptosystems. CACM 26(1):96–99.

[15] Elgamal T. A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans Inf Theory. 1985;31(4):469–472. doi: 10.1109/TIT.1985.1057074.

[16] Josh B. Dense probabilistic encryption. In Proceedings of the workshop on selected areas of cryptography, pp 120–128.

[17] Qian Lou, Wen-jie Lu, Cheng Hong, Lei Jiang. Falcon: Fast Spectral Inference on Encrypted Data. In: Proceedings of the CCS, 2020.