Numerical Methods with Fully Homomorphic Encryption: Demonstration of Crypto PDE Solver

Jordan Cotler and Semon Rezchikov

This is a set of running notes to orient the project. The notes are not meant to be sections of our future paper (although they may inspire sections in our future paper).

The Idea:

At present, fully homomorphic encryption (FHE) is not practical for general algorithms. The reason is that known FHE constructions work directly with arithmetic circuits, and scale unfavorably (polynomially, but not in a practical way) with the depth of the circuit that is to be encrypted. Most common algorithms represent very high-depth arithmetic circuits, and so are unsuitable for near-term FHE applications.

However, there is still opportunity to use FHE for tasks which require arithmetic circuits of modest depth. This situation occurs when the task in question is comprised of simple arithmetical operations. For instance, one can consider extracting basic statistical properties of a hidden data set.

A more sophisticated application which still yields a modest depth arithmetic circuit is the evaluation of a trained neural network [1]. This is similar in spirit to extracting statistical properties of a hidden data set, but for neural networks the extracted featured can be more complex. Indeed, one can think of a simple feedforward neural network as a (relatively) low-depth arithmetic approximation to a complicated function which may be difficult to describe *exactly*.

Here, we suggest a paradigm distinct from evaluating functions of data. Namely, applying numerical methods to dynamically evolve data, or to use data as an input to solve some numerical problem. The corpus of numerical methods used in engineering and the sciences is vast, and comprises mainly of algorithms which are naturally

expressed as arithmetic circuits. We anticipate that it will be essential to develop techniques to interface such numerical methods with FHE. For our work, we will focus on the example of simple evolution PDE's, such as the heat equation, to kick off this field of investigation. Our work has two main purposes:

1. To introduce, in a sharp way, the problem of interfacing numerical methods with FHE, as well as defining goals and benchmarks.

2. To work out a specific instance of this "programme," namely a PDE solver. We then demonstrate how it addresses our proposed goals and benchmarks.

Crypto PDEs Solvers

There are two main categories of PDE problems:

1. Evolution equations with initial conditions

2. Boundary value problems

We will focus on the former. PDE's require several inputs:

1. Initial conditions

2. Parameters which define the PDE

3. Time to evolve

Let us assume that (3) will not be encrypted. Otherwise, hiding the depth of an arithmetic circuit can become a nightmare. The two most plausible situations are that the client would want to hide either (1), or both (1) and (2).

If the client only wants to hide (1), then if the PDE is linear we only need to do partially homomorphic encryption (i.e., only for addition). Fast methods already exist for this. If the PDE is non-linear, then we need to use FHE. By contrast, if the client wants to hide both (1) and (2), then we need to use FHE regardless of whether or not the PDE is linear.

We will work with the 2D heat equation

$$\frac{\partial u}{\partial t} = \alpha \nabla^2 u \,, \qquad u(\mathbf{x}, 0) = u_0(\mathbf{x}) \tag{1}$$

where $u(\mathbf{x}, t) : \mathbb{R}^2 \times [0, T] \to \mathbb{R}_{\geq 0}$. We might also consider the non-homogeneous (and anisotropic) heat equation

$$\frac{\partial u}{\partial t} = \nabla \cdot A(\mathbf{x}) \nabla u \,, \qquad u(\mathbf{x}, 0) = u_0(\mathbf{x}) \tag{2}$$

for some function $A : \mathbb{R}^2 \to \mathbb{R}_{\geq 0}$.

Let us focus on the simpler case of Eqn. (1) for now. We work on a lattice L with N sites. A final time T is specified by the client, and the values of α and $u_0(\mathbf{x})$ (which here takes $L \to \mathbb{R}_{\geq 0}$) are to be hidden. Letting D be the discrete Laplacian matrix, we implement the update rule

$$u(\mathbf{x}, t) \longrightarrow u(\mathbf{x}, t + \epsilon) = (1 + \epsilon \, \alpha \, D) \, u(\mathbf{x}, t) \,. \tag{3}$$

We repeat the update rule T/ϵ times, starting from $t = 0$.

Two-party Protocols

If the depth gets too long, the server can send the output to the client. The client can decode and encode again, and send the data back for further computation.

Taking advantage of repetitious circuits?

PDE solver arithmetic circuits are highly repetitious. Can we take advantage of this somehow?

Parallelization

How to parallelize.

Tradeoff of Parameters

Ideas for parameter tradeoffs, and plots to make.

References

[1] Gilad-Bachrach, Ran, et al. "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy." *International Conference on Machine Learning*. 2016.

[2] Chen, Hao, Kim Laine, and Rachel Player. "Simple encrypted arithmetic library-SEAL v2.3.0." 2017.