GraphSC: Parallel Secure Computation Made Easy

This paper mainly focuses on 2-party secure computation in a semi-honest model where two non-colluding parties collaboratively perform secure computations over their joint secret graph data without data exposing. The key challenges are to propose a programming paradigm for non-experts in cryptography to perform secure computation of most graph-based algorithms with parallelism.

This paper proposes GraphSC, an easy and secure parallel programming paradigm for graph-based algorithms. Specifically, in order to provide such easy programming paradigm, the paper follows GraphLab and Pregel to use Scatter (propagate data), Gather (aggregate data), and Apply (computation) operations as three primitives in the framework so that most data mining and machine learning algorithms can be cast in this provided framework. Furthermore, in order to design secure parallel graph-based programming paradigm, authors firstly propose the parallel data oblivious algorithms using these three primitives where oblivious sort is integrated into Scatter and Gather, and aggregate operations and propagate operations are parallelized. Finally, standard techniques are used to convert data oblivious algorithms into secure algorithms.

Bucket Oblivious Sort: An Extremely Simple Oblivious Sort

This paper proposes simple and efficient oblivious sort and oblivious random permutation algorithms to protect against information leakage from the data access pattern to the outsourced data.

The basic idea is to design an oblivious random bin assignment algorithm to assign the elements into the bins randomly and evenly in an oblivious way, where a butterfly network is used to perform such assignment. The designed random bin assignment algorithm is oblivious since the execution is independent of the input data. After performing the oblivious random bin assignment algorithm on the data, we can use the non-oblivious comparison-based sort to achieve oblivious sorting and simply permute to achieve oblivious random permutation.