Amortized Sublinear Secure Multi Party Computation

Dov Gordon* Jonathan Katz† Vladimir Kolesnikov‡
Tal Malkin* Mariana Raykova* Yevgeniy Vahlis*

February 17, 2011

Abstract

We study the problem of secure two-party and multi-party computation in a setting where some of the participating parties hold very large inputs. Such settings increasingly appear when participants wish to securely query a database server, a typical situation in cloud related applications. Classic results in secure computation require work that grows linearly with the size of the input, while insecure versions of the same computation might require access to only a small number of database entries.

We present new secure MPC protocols that, in an amortized analysis, have only polylogarithmic overhead when compared with the work done in an insecure computation of the functionality. Our first protocol is generically constructed from any Oblivious RAM scheme and any secure computation protocol. The second protocol is optimized for secure two-party computation, and is based directly on basic cryptographic primitives.

1 Introduction

We propose a new mechanism for secure two party computation using a combination of Random Access Machines (RAM) and traditional MPC. RAMs [**?**] are a natural computational model, where an algorithm has direct access to the memory space.

As secure two party computation protocols mature, we begin to identify and address bottlenecks in performance that stem from fundamental properties of the current techniques. One of the main such bottlenecks is the fact that MPC is based on boolean or arithmetic circuits, rather than RAMs. Circuits, even if evaluated insecurely, must separately encode and process every possible evaluation path (e.g., both branches of `if` statements must be evaluated). In particular, the description of a boolean circuit must be at least as long as the input to the algorithm that the circuit encodes.

When computing on large inputs, e.g., when accessing a database, this is prohibitive. For example, any type of search on sorted DB will always require accessing all records, instead of the logarithmic cost of the (insecure) binary search. The state of affairs is that there are currently no satisfactory solutions for privately computing on large databases.

*Columbia University. {gordon,tal,mariana,evahlis}@cs.columbia.edu
†University of Maryland. jkatz@cs.umd.edu
‡Bell Labs. kolesnikov@research.bell-labs.com

A new approach for secure two party computation. We propose a new approach for secure computation where the functionality is represented as a RAM. We then design secure protocols in this model. The potential gains are:

- RAMs maintain a small state and access only the needed data. When inputs are large, this will lead to significant gains in performance. As an example, a secure binary search on a sorted database will require accessing roughly $O(\log^c n)$ entries (for some constant c), rather than reading the entire database, as is required in current MPC techniques.
- Many computationally intensive tasks admit algorithms that perform well in most situations, but can perform quite badly in the worst case. Using RAM as the basic model of computation will allow us to take advantage of the efficiency of such algorithms. Our secure computation protocols will only introduce cryptographic overhead for the operations performed by the insecure version of the algorithm. In contrast, existing protocols rely on circuit representations and always run (at least) at the worst-case running time of the algorithm (with additional cryptographic overhead).

At first glance, it may appear that sublinear secure computation is impossible. Indeed, consider computing $f(x, i) = x_i$ (i.e. f simply returns the ith bit of x). Then, unless the entire input x is accessed during evaluation of f, the party holding x will learn that $i \neq j$ for every j that is not accessed during the computation. We address this by adding a form of joint pre-processing on both inputs x and y, which will hide the location of the data. Importantly, we only require a *one-time* pre-processing, which will then allow the secure evaluation of an *unbounded* number of arbitrary functions on x and y in the future. Each such evaluation of the function f_i *requires only as much data access as does the insecure version of* f_i, with some fixed overhead that does not depend on the sizes of x and y.

This pre-processing can be done as part of the protocol, or, as would be pertinent in many applications, in advance by a supervising third party. An example is an encrypted fingerprint database where users, such as the FBI, have the appropriate key to access the data. The user can then perform a secure search for a specific fingerprint match without revealing the query to the server, and the server is able to answer the query without scanning the entire database, and without revealing any intermediate results.

Existing solutions for secure MPC using RAM representation. Several approaches exist (e.g. [**?**, **?**, **?**, **?**]) for securely evaluating functionalities that are represented as random access machines (rather than boolean or arithmetic circuits). Indeed, when possible, these solutions achieve communication complexity sublinear in the sizes of the inputs. However, to our knowledge, no secure MPC protocols exist where the *computational work* done by the parties is sublinear in the input size.

Our approach. We propose an approach for securely evaluating a two party function $f(x, y)$ based on its RAM representation. One of the basic tools that we use are Oblivious RAM protocols [**?**, **?**, **?**, **?**]. Such protocols allow a client holding a key K to evaluate an algorithm represented as RAM on data which is encrypted using K, and stored on a server. During the protocol, the server learns nothing about the data, or the *data access pattern* of the client. Moreover, the client only needs to store a small state, whose size does not depend on the size of the data that is stored on the server.

The main mismatch between the goals of Oblivious RAM and secure two-party computation is that the client in the former is trusted, and in particular learns intermediate values during the computation (e.g. for binary search, the client would learn the values on the entire path to the requested entry). We therefore propose to combine Oblivious RAM techniques with secure two-party computation to gain security against both parties. In more detail:

1. The two parties, holding inputs x and y respectively, run a pre-processing protocol, where each party obtains an encrypted version of its own input under a key that is given to the other party. As a result, the first party holds a key K_1 and an encryption of x under K_2, and the second party holds K_2 and encryption of y under K_1.

2. After the initial pre-processing phase, the two parties can repeat the following process an unbounded number of times. Given a two-party functionality $f(x, y)$, the parties generate an encoding $\hat{f}$ of f as a RAM according to the specification of the Oblivious RAM protocol. They then engage in a sequence of two party computation protocol instances, jointly evaluating the "next instruction" function of $\hat{f}$ on their respective inputs. The state of the machine $\hat{f}$ is secret-shared between the two parties. That is, the two parties act both as the client and the server in the oblivious RAM protocol by secret-sharing the state of the RAM between them, and jointly computing the next state via a secure two-party computation protocol.

The key idea is that since the access pattern is hidden in Oblivious RAM, the accessed locations of the encoded x and y can be made public! This allows us to run the MPC protocol only on the parts of x and y that are needed for the evaluation of $\hat{f}$, and leave the rest of the input untouched. Note, we do not need to apply Oblivious RAM techniques to the party whose input is short (resulting in corresponding simplification and performance improvement). This is the case, for example, when searching large databases.

One subtlety that arises in this model is that the runtime of the protocol might depend on the particular inputs of the parties. This is not the case when we represent the computation as a circuit, since the circuit size is independent of the particular input being used. As part of the model design, we allow the adversary to gain information leaked by the running time of RAM. Note, in many functions, e.g. in search, the runtime can be cheaply padded to the logarithmic bound, and will reveal nothing. When input-based shortcuts are taken, this necessarily reveals runtime, so in a sense, this small compromise is unavoidable. After developing the model and the above techniques we extend them to multiple parties, and to malicious adversaries.

Efficiency improvements. The above construction is completely generic, and would work with any secure two-party computation protocol, and any Oblivious RAM protocol. The main drawback of the generic solution is that, depending on the ORAM protocol, the functionalities evaluated by the MPC protocol may be quite complex. This in turn would reduce the usability of the Secure-RAM protocol. The second part of our work consists of a new Secure-RAM protocol, designed from scratch for practical efficiency. Our construction is based on ideas from the Pinkas-Reinman oblivious RAM protocol and standard garbled circuit techniques. The resulting protocol is quite efficient, and the cryptographic functionalities that are evaluated using secure MPC are quite simple. We expect that our second protocol will be able to handle input sizes that are beyond reach for current MPC implementations.