2.4 Secure Multi-Party Computation

As a brief background, we introduce the basic ideas of secure multi-party computation that are used in this work.

The idea behind SMPC is to securely compute a function by dividing the computation across multiple parties such that no single party can inspect intermediate and sensitive parts of the computation until all parties agree to reveal the results. Thus, the parties can jointly compute a function on their private data without having to aggregate their data in the open.

The typical procedure in cryptography is to first declare the assumptions of what information can be learned from the joint computation a priori and what knowledge is already assumed prior to the computation. Whatever can be learned from the output after the computation is irrelevant to the security that the cryptosystem provides. The system only guarantees that the system – the computation – itself, in addition to the assumptions, will not allow anything to be learned during the computation.

SMPC is used when local computation is more expensive then communication cost.

2.4.1 Additive Secrete Sharing

The most introductory technique in SMPC is additive secrete sharing, which aims to compute any function only involving additions. A typical example would be if multiple parties in a company were interested in computing the average of their salaries without revealing their salaries to each other. Assume for this example, there are three parties – commonly this scenario with three parties is referred to 3pc (three-party computation). The first party generates a one-time pad, which is a string generated from a uniform distribution, adds it to their salary, and sends it to the second party. The second party does the same, adding their salary with their own one-time pad and the random string they received from the first party. This second party then sends this new random string to the third party. The third party repeats the process and the passes the new random string to the first party. This first phase is called the masking phase. In the second phase, referred to as the unmasking phase, the first party now subtracts their one-time pad from the random number and passes it to the next party. The unmasked sum eventually returns to the first party. The sum is then passed around to each party so that each party has a copy. The parties can now individually do as they please with the system and the guarantees of the cryptosystem end here. Each party can now calculate the average by dividing by three and infer whatever information they can from this output. One observation that a party can make is how their input compares to the average. If their input is larger than the average, then they know that at least of the of the other two parties has a salary that is less than theirs. Additionally, depending on the difference between the average and their input, a party can determine whether just one or both parties have a salary that is less than their own. If this party has additional outside knowledge, they could possibly approximate or even exactly determine what the salaries were. At this point, it may sound counterproductive to even jointly compute with other parties, but no cryptosystem can guarantee an absolute defense and cannot make any guarantees outside of the cryptosystem. Here, the communication cost nine rounds of communication to securely compute the sum and share it with all parties.

//insert diagram here

2.4.2 Secure Multi-Party Multiplication

Securely multiplying numbers requires more work than addition. First, the numbers must be integers and second the numbers must be secret shared across parties. As an example, assume three parties want to multiply . This example can be extended to n parties by secrete sharing with n parties.

Securely multiplying requires more computation. Generally, as the operations required for a computation become more complex, more communication is required to compute it securely.

//illustrate integer multiplication and stages of computation among parties

2.4.3 Secure Multi-Party Matrix Multiplication

Building on then previous section, multiple parties can securely multiply matrices by using the secure multiplication primitives for each multiplication in a matrix. One can imagine that the cost to compute one matrix multiplication is prohibitively expensive. Requiring x-communications for n parties. For this reason, secure matrix multiplication was not used in this work. However, we mention it for completeness for the reader.

//illustrate secrete sharing matrix and stages of computation