Errata to MPC Course

Naman Kumar

March 2, 2024

Abstract

Basic notes on MPC.

Contents

1 Secure N-party Computation

It's probably worth reiterating some of the formalisms of these definitions with a bit more lucidity, just as a simple reference and as some sort of illumination.

1.1 Real World Instantiation

In the real world, N parties have inputs $\mathbf{x} = (x_1, \ldots, x_n)$, an agreed-upon function $f : (x_1, \ldots, x_n) \mapsto (y_1, \ldots, y_n)$ and they follow a protocol Π which gives them the set of *party outputs* $\text{OUT}_\Pi(\lambda, \mathbf{x}) := (y_1, \ldots, y_n)$. Assume that t of these parties are controlled by the adversary; define the *view* $\text{VIEW}_{\Pi,\mathcal{A}}(\lambda, \mathbf{x})$ to be the set of all inputs of corrupted parties, along with messages exchanged that the adversary has sent, received, or eavesdropped-upon.

Definition 1.1 (Real World Distribution)**.** *The real-world output* $\text{REAL}_{\Pi,\mathcal{A}}(\lambda, \mathbf{x})$ *is defined as the tuple*

$$\text{REAL}_{\Pi,\mathcal{A}}(\lambda, \mathbf{x}) := (\text{OUT}_\Pi(\lambda, \mathbf{x}), \text{VIEW}_{\Pi,\mathcal{A}}(\lambda, \mathbf{x})).$$

1.2 Ideal World Instantiation

In the ideal world, there is no protocol, only a functionality $\mathcal{F}$ which takes in inputs from each party, computes the output f, and returns $(f(x_1), \ldots, f(x_n))$ to the parties. The output, $\text{OUT}_F(\lambda, \mathbf{x})$ is the same – the output of the parties after the protocol execution – while instead of the *view* (which is, of course, not identical to that of the actual ideal-world 'protocol' execution; a protocol could be multi-round), there is the *view of a simulator* (a 'fake' adversary which works in the ideal world, but has access to the inputs of the real adversary), which is the output of the simulator on any given execution.

Definition 1.2 (Ideal World Distribution)**.** *The ideal-world output* $\text{IDEAL}_{\mathcal{F},\mathcal{S}}(\lambda, \mathbf{x})$ *is defined as the tuple*

$$\text{IDEAL}_{\mathcal{F},\mathcal{S}}(\lambda, \mathbf{x}) := (\text{OUT}_\mathcal{F}(\lambda, \mathbf{x}), \text{VIEW}_{\mathcal{F},\mathcal{S}}(\lambda, \mathbf{x})).$$

Note that these distributions are *joint* distributions.

1.3 Secure N-Party Protocol

Definition 1.3 (t-Privacy of a Protocol)**.** *An N-Party protocol Π is considered t-private if for any PPT adversary that corrupts t of the parties, there exists a PPT simulator such that*

$$\{\text{IDEAL}_{\mathcal{F},\mathcal{S}}(\lambda, \mathbf{x})\} \equiv \{\text{REAL}_{\Pi,\mathcal{A}}(\lambda, \mathbf{x})\}.$$

Remark. There's two examples here of the non-privacy of a protocol. One of them is a function which takes no inputs and outputs a random $b \leftarrow \{0, 1\}$ to one of the parties,

say P_0. The protocol in which P_1 samples a bit and sends it to P_0 is not secure since if P_1 is corrupted, the view of the adversary is different; it has the bit b in it. Similarly, if a functionality takes no input and outputs pq, $(p+q)$ to the parties, then, again, the protocol in which P_0 chooses p, q and sends $p+q$ is insecure since it learns the numbers p and q.

2 Oblivious Transfer

Oblivious Transfer is a simple MPC functionality parametrized by a selection of sender inputs and a receiver index.

Parameters: The sender S has a selection of N input strings $(m_0, \ldots, m_{N-1})$, while the receiver R has an index $i \in [N]$.
Outputs: S receives nothing while R receives m_i.

Figure 1: 1-out-of-N Oblivious Transfer.

2.1 Protocol for 1-out-of-2 OT

We now demonstrate a simple protocol for 1-out-of-2 OT [EGL85]. We begin with a CPA-secure encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, and define the notion of oblivious sampling.

Definition 2.1 (PKE with Obliviously Sampleable Encryption Key)**.** *An encryption scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *has obliviously sampleable public keys if there exist algorithms* Samp *and* pkSamp *such that*

- $\{\mathsf{Samp}(1^\lambda)\}$ *is computationally indistinguishable from* $\{\mathsf{pk} : (\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)\}$.
- $\{(\mathsf{pk}, r) : r \xleftarrow{\$} \{0,1\}^\lambda, \mathsf{pk} \leftarrow \mathsf{Samp}(1^\lambda; r)\}$ *is computationally indistinguishable from* $\{(\mathsf{pk}, r) : (\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Gen}, r \leftarrow \mathsf{pkSim}(\mathsf{pk})\}$.

The protocol proceeds as follows.

- Receiver runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and $\mathsf{pk}' \leftarrow \mathsf{Samp}(1^\lambda)$ and sets $\mathsf{pk}_b = \mathsf{pk}$, and $\mathsf{pk}_{1-b} = \mathsf{pk}'$.
- Receiver sends pk_0 and pk_1.
- Sender encrypts $c_i = \mathsf{Enc}_{\mathsf{pk}_i}(m_i)$.
- Sender sends c_0, c_1.
- Receiver decrypts $m_b = \mathsf{Dec}_{\mathsf{sk}_b}(c_b)$.

Figure 2: 1-out-of-2 Oblivious Transfer from obliviously sampleable PKE.

3 Modifications to OT

References

[EGL85] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, jun 1985.