Additional MPC Exercises

Jesper Buus Nielsen
University of Aarhus,
buus@daimi.aau.dk

May 11, 2005

1 Additional MPC Exercises

1.1 (Im)Possibility of security $t \geq n/2$

We proved in Section 4.1 that for all functions f there exists a protocol which perfect securely realizes f against a passive adversary which corrupts $t < n/2$ out of the n parties. We will later prove that this is optimal in the sense that it is not the case that for all functions f there exists a protocol which perfect securely realizes f against a passive adversary which corrupts $t = n/2$ parties. This does however not excluded that there exists *some* functions which can be securely realized against passive adversaries corrupting more than $n/2$ parties.

Exercise 1 Find a class of functions f for which you can realize all of them against a passive adversary corrupting up to $t = n - 1$ out of the n parties. [Hint: Have a look at the protocol in Section 4.1 and identify where it was used that $t < n/2$.]

1.2 General Adversary Structures

We have until now looked at secret sharing schemes secure against a passive cheating. A stronger notion of security is so-called crash security. Here corrupted parties might crash and thereby stop sending messages. However, until a party crashes it behaves honestly.

Let Γ denote an adversary structure over $[n] = \{1, \ldots, n\}$. I.e. $C \in \Gamma$ means that $C \subset [n]$ and that the adversary is allowed to corrupt the parties P_i for $i \in C$.

A Γ-crash adversary is an adversary which might crash parties P_i for $i \in C$ as long as $C \in \Gamma$.

Assume now that we want a secret sharing scheme which is secure against a Γ-crash adversary, where security means that 1) the secret sharing scheme leaks no information to a subset of parties P_i for $i \in C$ when $C \in \Gamma$ and that 2) even if a subset $C \in \Gamma$ crashes, the remaining parties can still reconstruct the secret. The first requirement is called privacy:

Definition 1 *A secret sharing scheme has* Γ-privacy *if for all* $C \in \Gamma$ *it holds that if* $s_1, \ldots, s_n$ *is a random secret sharing of some secret* s*, then* $\{s_i\}_{i \in C}$ *leaks no information about* s*.*

We formalise the second requirement.

If we secret share a secret s among the n parties using shares $s_1, \ldots, s_n$ and if all parties P_i for $i \in C$ are corrupted, then we might risk that we only get the shares $s_i \in [n] \setminus C$ when reconstructing, as the parties P_i for $i \in C$ might crash before the reconstruction.

For any set C we let $\bar{C} = [n] \setminus C$ and for an adversary structure Γ we let $\bar{\Gamma} = \{\bar{C} | C \in \Gamma\}$. As mentioned above, when doing secret sharing in context of a Γ-crash adversary, our only

guarantee in the reconstruction is to get s_i for $i \in H$ for some $H \in \bar{\Gamma}$. This motivates the following definition.

Definition 2 *A secret sharing scheme has* Γ-crash reconstruction *if for all* $H \in \bar{\Gamma}$ *and all secrets* s *and all sharings* $s_1, \ldots, s_n$ *of* s *it holds that* s *can be computed from* $\{s_i | i \in H\}$.

Exercise 2 Argue that if a secret sharing scheme has both Γ-privacy and Γ-crash reconstruction, then Γ is Q2.

1.3 (Im)Possibility of security $t \geq n/2$

In the MPC note it is argued that the corruption bound $t < n/2$ is optimal by proving that there exists two-party functions $z = f(x, y)$ which cannot be computed perfectly securely if one of the parties is allowed to be passively corrupted (here $z = f(x, y)$ denotes the function $(z, z) = f(x, y)$, where both parties learn the same output).

This is argued by proving that this is in particular the case for the AND function $c = a \wedge b$ where $a, b \in \{0, 1\}$ and $a \wedge b = 1$ iff $a = 1$ and $b = 1$.

Having established this result for the AND function we can easily get it for any function which 'can be used to securely compute an AND'. Consider e.g. the MULT function $z = x \cdot y$ where e.g. $x, y \in \{0, \ldots, 100\}$ and $z \in \{0, \ldots, 10000\}$. This function cannot be computed secure. Namely, if it could then we could also compute the AND function securely, a contradiction. To see this consider the following protocol for the AND function: The parties P_1 and P_2 have inputs $a, b \in \{0, 1\}$ respectively. Then they run the MULT protocol on inputs $x = a$ and $y = b$ respectively, to learn the value $z = a \cdot b$, and nothing else. They learn the correct result, as $a \cdot b = a \wedge b$ when $a, b \in \{0, 1\}$, and the protocol is secure as they *only* learn $a \cdot b = a \wedge b$.

We say that the AND function can be securely reduced to the MULT function. In general we say that AND can be securely reduced to a function $z = f(x, y)$ if there exist six values $x_0, x_1, y_0, y_1, z_0, z_1$ such that $f(x_a, y_b) = y_{a \wedge b}$ for all $a, b \in \{0, 1\}$.

If AND can be securely reduced to f, then a protocol for securely computing f allows to securely compute AND, as follows: Party P_1 gets input a and then inputs x_a to the protocol for securely computing f. Party P_2 gets input b and then inputs y_b to the protocol for securely computing f. Then they learn the output $z = f(x_a, y_b)$, and noting else. Then they check whether $z = z_0$ or $z = z_1$ and outputs 0 respectively 1.

It therefore follows that if AND can be securely reduced to a function $z = f(x, y)$, then f cannot be computed perfectly securely if one of the parties is allowed to be passively corrupted.

Exercise 3 Show that the following functions cannot be computed perfectly securely if one of the parties is allowed to be passively corrupted.

1. The OR function $(x, y) \mapsto x \vee y$, where $x, y, z \in \{0, 1\}$ and $0 \vee 0 = 0$ and $x \vee y = 1$ otherwise. [Hint: $x_0 = 1, x_1 = 0, y_0 = 1, y_1 = 0, z_0 = 1, z_1 = 0$.]

2. Any binary Boolean function $z = f(x, y)$, where $x, y, z \in \{0, 1\}$ and $f(x, y) = 1$ for exactly 1 or 3 choices of $(x, y) \in \{0, 1\} \times \{0, 1\}$.

3. The millionaire's problem: The parties have inputs $x, y \in \{0, \ldots, B - 1\}$ for some integer B, $B = 1000000000$ say, and they want to compute $z = (x \stackrel{?}{<} y)$, where $z \in \{0, 1\}$ and $z = 1$ iff $x < y$.

4. The string-matching problem: The parties have inputs $x, y \in \{0, 1\}^k$ for $k > 1$ and want to compute $z = (x \stackrel{?}{=} y)$, where $z \in \{0, 1\}$ and $z = 1$ iff $x = y$.

Exercise 4 Show that the following function *can* be computed perfectly securely even if one of the parties is allowed to be passively corrupted.

1. Then XOR function for two parties. I.e. the function $z = x \oplus y$, where $x, y, z \in \{0, 1\}$ and $z = x + y \bmod 2$. [Hint: Prove it by giving a secure protocol — there is a very simple protocol where each parties sends exactly one bit! You don't need to prove the security formally, just argue that it is correct (computes the XOR) and private (no party learns more than it can learn from its own input and the output).]

2. The string-matching problem with $k = 1$.

3. Any binary Boolean function $z = f(x, y)$, where $x, y, z \in \{0, 1\}$ and $f(x, y) = 1$ for exactly 0, 2 or 4 choices of $(x, y) \in \{0, 1\} \times \{0, 1\}$.

1.4 General Adversary Structures

Let Γ denote an adversary structure over $[n] = \{1, \ldots, n\}$. I.e. $C \in \Gamma$ means that $C \subset [n]$ and that the adversary is allowed to corrupt the parties P_i for $i \in C$.

We can construct a secret sharing scheme with Γ-reconstruction by sharing the secret s among all subsets which should be able to reconstruct, i.e. every $H \in \bar{C}$, or equivalently, every $H = [n] \setminus C$ for $C \in \Gamma$.

Specifically, assume that $s \in G$ for some finite Abelian group G, which we write with additive notation. If s is a number $s \in \{0, \ldots, p-1\}$ we can simply let $G = \mathbf{Z}_p$ with addition modulo p. The dealing phase of the secret sharing scheme then proceeds as follows:

1. The input is a secret $s \in G$.

2. For every $C \in \Gamma$ do the following:

   (a) Let $H = [n] \setminus C$.
   (b) For each $i \in H$ pick a uniformly random element $s_{H,i} \in_R G$ with the only condition that $\sum_{i \in H} s_{H,i} = s$. (where the sum is taken in G).
   (c) For each $i \in G$, send $s_{H,i}$ securely to P_i.

3. The share of P_i is thus of the form $s_i = (s_{H_1,i}, \ldots, s_{H_m,i}) \in G^m$ where $H_1, \ldots, H_m$ are the sets $H \in \bar{\Gamma}$ with $i \in H$.

Let us call this *the generic linear secret sharing scheme.*

The drawback of this scheme is that it can be very inefficient, as each P_i gets a values $s_{H,i}$ for each $H \in \bar{\Gamma}$ for which $i \in H$. The advantage is that it is simple, and optimally secure, where by optimally secure we mean that it has Γ-crash reconstruction and that it has Γ-privacy if Γ is Q2. This is optimal in the sense that we know from Exercise 2 that if Γ is not Q2, then there is no secret sharing scheme which has Γ-crash reconstruction and at the same time has Γ-privacy.

Exercise 5 Show that the generic linear secret sharing scheme has Γ-crash reconstruction. I.e. the honest parties can always reconstruct without the shares of the corrupted parties.

Exercise 6 Show that if Γ is Q2, then the generic linear secret sharing scheme has Γ-privacy.

Exercise 7 Show that the generic linear secret sharing scheme is linear. I.e. if $(s_1, \ldots, s_n)$ is a sharing of s and $(u_1, \ldots, u_n)$ is a sharing of u, then $(s_1 + u_1, \ldots, s_n + u_n)$ is a sharing of $s + u$. Here s_i and u_i are vectors of elements from G, and addition denotes pointwise addition of these vectors in G.