1 Private Data Collection

Secure aggregation. Suppose we have a large number of clients, each holding x_i. The server wants to compute $\sum_i x_i$. Assume all clients have Diffie-Helman (DH) public key $h_i = g^{s_i}$, and each pair of clients i, j can compute shared key $k_{i,j}$. The the following is a basic protocol.

Protocol 1 (Basic Private Data Collection)**.** Each client sends to the server

$$y_i = x_i + \sum_{i<j} k_{i,j} - \sum_{i>j} k_{i,j},$$

where $\sum_{i<j} k_{i,j} - \sum_{i>j} k_{i,j}$ is $mask_i$, such that

$$\sum_i mask_i = 0.$$

This yields that

$$\sum y_i = \sum x_i.$$

Problem: Fragility. If the server fails to receive a single y_i value, then the entire protocol breaks down. It is very fragile!

Solution: Recovery Step. Consider this modification to the previous protocol.

1. Each client shares its DH secret s_i with other clients using a t-out-of-n secret sharing scheme.
2. The server can request shares of s_i for any y_i value it did not receive.

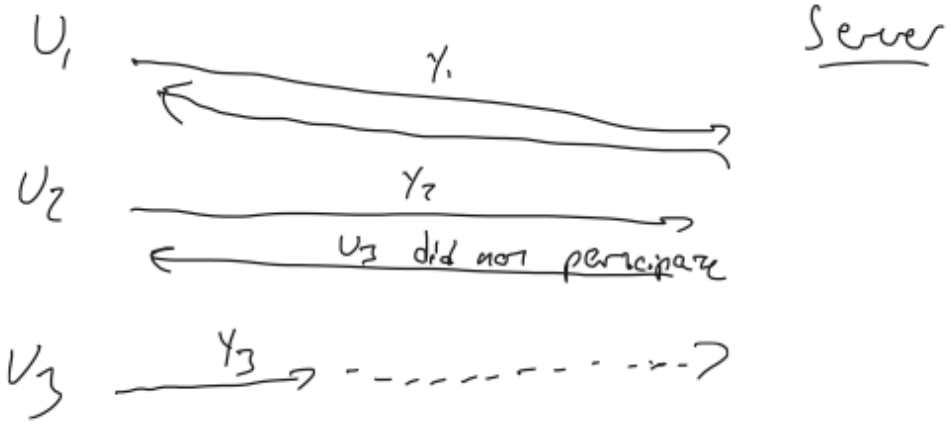

Figure 1: The recovery step modification to the basic private data collection protocol

Problem: Requires Semihonesty. Malicious users can break privacy of users, and delayed messages also break privacy.

Solution: Individual Masks Users can apply individual mass via

$$y_i = x_i + \sum_{i<j} k_{i,j} - \sum_{i>j} k_{i,j} + r_i$$

where users also secret share r_i among the other clients.

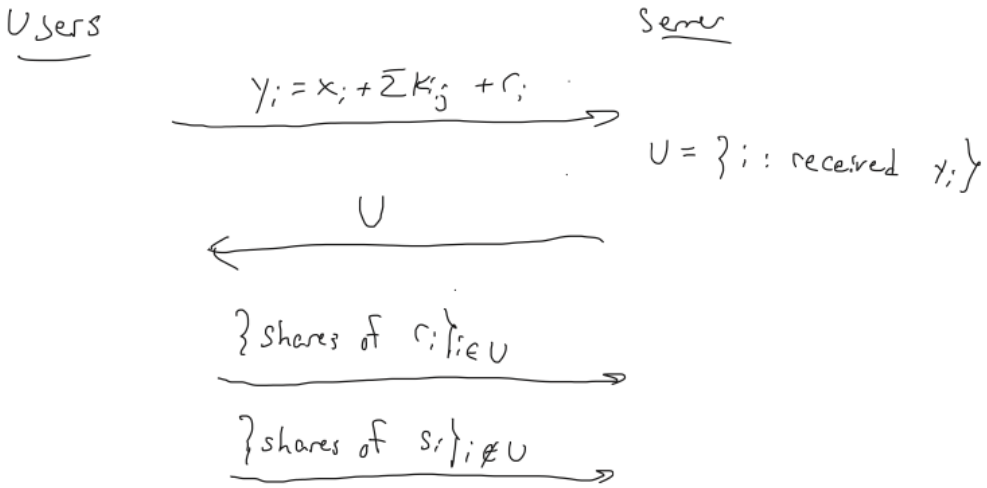

Figure 2: The individual masks modification to the basic private data collection protocol

1.1 Prio

Protocol 2 (Prio)**.** The context is: many clients, $1 < n$servers, secure against $n-1$ semi-honest servers, each client has input x_i, and we want to compute $\sum x_i$. The simple protocol is:

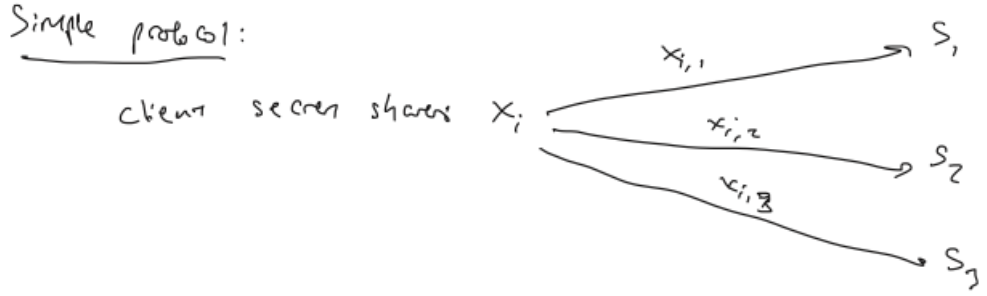

Figure 3: The simple version of the Prio protocol

To extend this simple protocol, the challenge that needs to be addressed is that we need to enforce that the clients' inputs satisfy some predicate i.e. $C(x) = 1$ for some predicate C. To accomplish this, there are three options.

1.1.1 Prio Protocol Extension 1

The server can run an MPC protocol evaluating $C(x)$ and check if the result is 1.

1.1.2 Prio Protocol Extension 2

The client can locally evaluate $C(x)$, and determine the value on every wire of the circuit. Then servers can verify correctness by secretely evaluating a $O(1)$-depth circuit.

1.1.3 Prio Protocol Extension 3

Client-side. The goal is to evaluate $C(x)$. Let M be the number of multiplication gates in C.

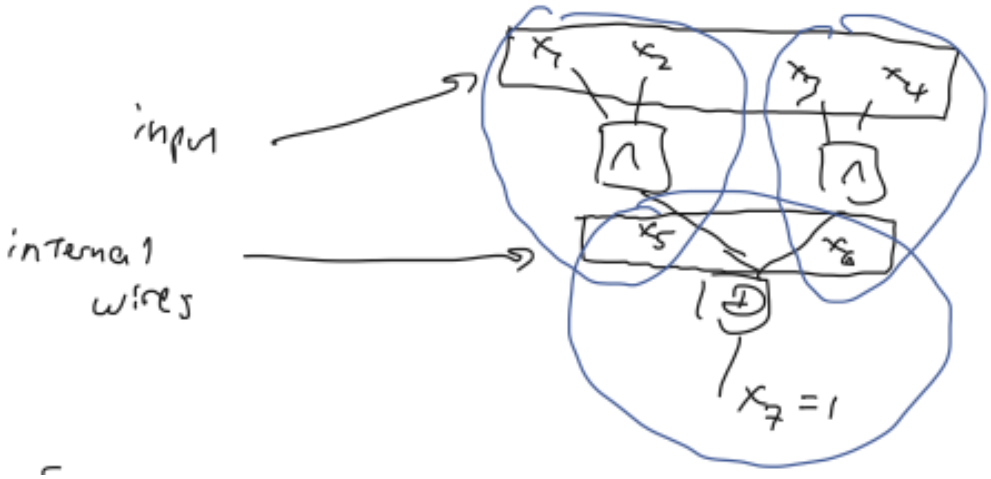

Figure 4: Option 2 for extending the simple Prio protocol

1. Let f be a polynomial such that $f(t) =$ the value on left input wire of gate t, for $1 \leqslant t \leqslant M$.
2. Let g be a polynomial such that $g(t) =$ the value on right input wire of gate t, for $1 \leqslant t \leqslant M$.
3. Let $h = f \cdot g$ i.e. $h(t) =$ the value of output wire of gate t.
4. Provide $[x], [f(0)], [g(0)], [h]$

Server-side.

1. Generate $[\hat{f}]$ and $[\hat{g}]$ locally.

$$f_i = \sum_{j=0}^{M} f(j) \cdot \gamma_j$$

$$[f_i] = \sum_{j=0}^{M} [f(j)] \cdot \gamma_j$$

$$h(x) = \sum h_i x^i \qquad [h(x)] = \sum x^i [h_i]$$

The claim two-fold:

1. If a client is honest, then $\hat{f} = f, \hat{g} = g, \hat{f} \cdot \hat{g} = h$
2. If h that a client shares is incorrect, then $\hat{f} \cdot \hat{g} \neq h$

So, the servers check whether

$$P(t) = t \cdot (\hat{f}(t) - \hat{g}(t) - h(t)) \equiv 0$$

i.e. they check if $P(r) = 0$ at a random r. If $P(t) \neq 0$, then

$$\mathsf{Pr}_r[P(r) = 0] \leqslant \frac{\deg(P)}{\mathbb{F}}$$

Some more details:

- servers agree on r
- each server locally computes $[r \cdot \hat{h}(r)], [\hat{f}(r)], [r \cdot \hat{g}(r)]$

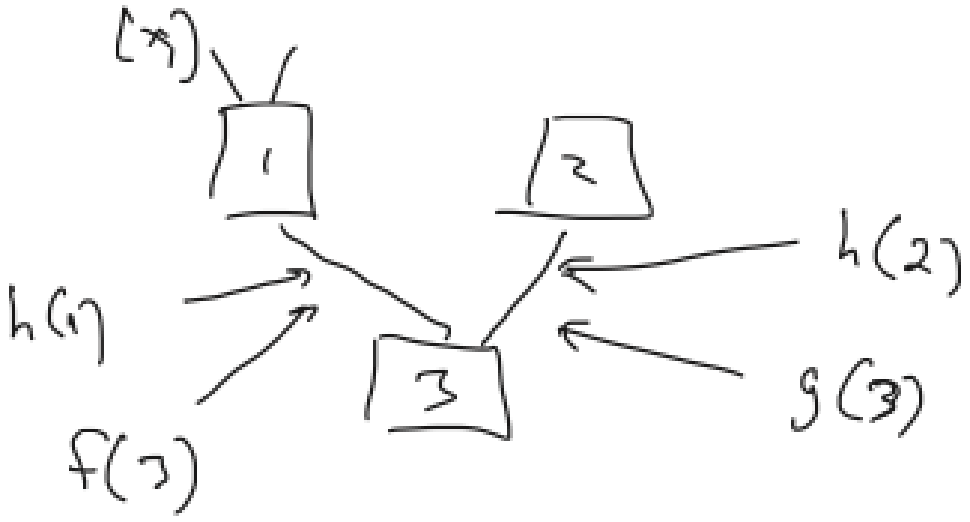

Figure 5: Example of calculations for Prio extension 3 server-side

- each client will also share Beaver triple $[a], [b], [c], c = ab + \delta$
- servers use Beaver triple to compute $[r \cdot \hat{f}(r) \cdot \hat{g}(r) + \delta]$
- servers check that $[r(\hat{f}(r) \cdot \hat{g}(r) - h(r))] = [0]$ and that $t \cdot \hat{f}(t) \cdot \hat{g}(t) - h(t)) + \delta = 0$.